











Library and Archives  
Canada

Published Heritage  
Branch

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque et  
Archives Canada

Direction du  
Patrimoine de l'édition

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file* *Votre référence*  
ISBN: 978-0-494-55269-8  
*Our file* *Notre référence*  
ISBN: 978-0-494-55269-8

#### NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

#### AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

  
**Canada**



# **Automated Recognition of Harp Seal Pups in Aerial Photographs**

by

© Jennifer Woodrow, B.Sc

A thesis submitted to the  
School of Graduate Studies  
in partial fulfilment of the  
requirements for the degree of  
Master of Science

Department of Computational Science  
Memorial University of Newfoundland

May 2008

St. John's

Newfoundland

## Abstract

Harp seals, *Pagophilus groenlandicus*, are the most abundant pinniped in the Northwest Atlantic. The Canadian and Greenland hunt for the harp seal is the largest marine mammal harvest in the world. To ensure a sustainable yield, it is important to monitor abundance and population trends on a regular basis. In addition to species management, assessing harp seal population is important in estimating the consumption of prey by the species. To estimate the total population of harp seals, the Canadian Department of Fisheries and Oceans (DFO) uses a population model that combines pup production estimates, pregnancy rates, and age-structured removals.

Currently, the number of harp seal pups are estimated by conducting visual and photographic aerial surveys over whelping concentrations. A fixed-wing aircraft, equipped with a large format metric mapping camera with motion compensation, is used to take black-and-white photographs of whelping areas. To count seal pups, manual analysis of aerial photographs is performed by trained scientific personnel with extensive knowledge of harp seals and their environment. This process can take many months and involve several people. While extensive measures are taken to ensure the most accurate pup count, manual identification of seal pups is not always conclusive. This thesis attempts to address these issues by developing image processing and pattern recognition tools that automatically detect and classify harp seal pups in digitized aerial photographs. Automating this process will reduce the amount of time required to compute population estimates and potentially improve the accuracy of pup counts.

The first step in the pattern recognition algorithm is to divide the large digitized aerial images into several sub-images for further analysis by the image processing and classification tools. The objective of the image processing algorithm is to segment and isolate potential harp seal objects to be used in pattern classification. The rigorous image processing component uses a combination of techniques including contrast stretching, adaptive thresholding using between-class variance, and a “cleaning” algorithm that employs edge detection, line dissection, and removal of objects based on size constraints. In addition, this thesis proposes a unique segmentation procedure called Isolate Connected Components that separates connected objects with minimal distortion to object shape.

The image processing routine calculates nineteen features for each segmented object. Features are optimized using three different methods: scaling the data, Principal Component Analysis, and kernel whitening. One-class classification methods use these features to identify an object as ‘seal pup’ or ‘not seal pup’. Two one-class methods are considered in this research: Parzen density estimation and Support Vector Data Description (SVDD). Optimal classifier parameters are determined by maximizing the Area Under the Receiver Operating Characteristic Curve (AUC). It is shown that the Parzen method performs better than the SVDD with an 82% success rate on test data.



## Acknowledgements

Completing my Masters degree has been a long and arduous process. I owe a debt of gratitude to a number of people for supporting me throughout the last few years. First and foremost, I would like to thank my supervisor Dr. George Mann for his constant motivation, feedback, and understanding of my chaotic schedule with work and school. I would not have completed this thesis if it weren't for his unwavering encouragement. Many thanks also to my co-supervisors (present and past) Dr. Serpil Kocabiyik and Dr. Richard Charron for their guidance and support throughout my graduate program.

Special thanks to Dr. Garry Stenson at the Department of Fisheries and Oceans (DFO, Newfoundland and Labrador region) for initiating this project and for providing valuable information about current techniques for seal population assessment. I also owe a tremendous thanks to Lori Hogan for providing the ground-truth images and digital image crops for my thesis work.

Thanks to Dilan Amarasinghe and Jason Mills for their assistance with Latex and thesis formatting.

I am forever grateful to my friends and co-workers Niki Legge and Mary Lynn Pender for providing a listening ear, for being flexible with scheduling, and for picking up the slack during my leave of absence. Also, thanks to Mary Lynn for reading my thesis and providing grammatical advice.

Last, but certainly not least, I would like to thank my family: my parents Richard and Marie; my sister Janine; and my puppy pals Curly, Willow, and Fozzie. I could not have done this without their love and support.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Statement . . . . .	1
1.2	Research Objectives . . . . .	2
1.3	Thesis Contributions . . . . .	3
1.4	Thesis Organization . . . . .	4
<b>2</b>	<b>Background</b>	<b>6</b>
2.1	Harp Seals . . . . .	6
2.1.1	Population Assessment . . . . .	8
2.1.2	Aerial Visual and Photographic Surveys . . . . .	10
2.1.3	Manual Photograph Analysis . . . . .	11
2.2	Previous Work . . . . .	14
2.3	The Approach . . . . .	21
<b>3</b>	<b>Image Segmentation</b>	<b>25</b>
3.1	Introduction . . . . .	26
3.2	Data Set . . . . .	27
3.2.1	Data Acquisition . . . . .	27
3.2.2	Data Digitization and Reduction . . . . .	28

3.2.3	Challenges . . . . .	32
3.3	Segmentation Algorithm . . . . .	35
3.3.1	Image Enhancement . . . . .	35
3.3.2	Adaptive Thresholding and Between-Class Variance . . . . .	38
3.3.3	“Cleaning” Algorithm . . . . .	48
3.3.4	Isolate Connected Components . . . . .	51
3.3.5	Remove Outliers . . . . .	60
3.4	Results . . . . .	61
3.5	Feature Selection . . . . .	65
<b>4</b>	<b>Classification</b>	<b>70</b>
4.1	Introduction . . . . .	71
4.1.1	Classifier Design and Performance . . . . .	74
4.2	One-Class Classification . . . . .	80
4.2.1	Error Definitions . . . . .	80
4.2.2	Distance/Resemblance Threshold . . . . .	84
4.2.3	ROC Curve . . . . .	85
4.2.4	One-Class Methods . . . . .	87
4.3	Parzen Density Estimation . . . . .	89
4.4	Support Vector Data Description . . . . .	90
4.4.1	Derivation of SVDD . . . . .	90
4.4.2	The Kernel Trick . . . . .	95
4.5	Experimental Results . . . . .	97
4.5.1	Feature Optimization . . . . .	98
4.5.2	Methodology for Training and Testing . . . . .	104
4.5.3	Final Results . . . . .	107

<b>5</b>	<b>Conclusions and Recommendations</b>	<b>112</b>
5.1	Summary of Research Methods . . . . .	112
5.2	Summary of Results and Conclusions . . . . .	113
5.3	Recommendations . . . . .	117
<b>A</b>	<b>Pseudocode for ICC Algorithm</b>	<b>133</b>
<b>B</b>	<b>Performance of Segmentation Algorithm on Training Data</b>	<b>135</b>
B.1	Excellent Performance . . . . .	135
B.2	Good Performance . . . . .	136
B.3	Satisfactory Performance . . . . .	138
B.4	Poor Performance . . . . .	139
B.5	Failure . . . . .	140
<b>C</b>	<b>Hu Moments</b>	<b>143</b>
<b>D</b>	<b>Classification Theory</b>	<b>144</b>
D.1	Error Definitions . . . . .	144
D.2	SVDD with Negative Examples . . . . .	144
D.3	Kernel Whitening . . . . .	146
D.3.1	Chernoff Distance . . . . .	148

# List of Tables

2.1	Summary of automated population assessments . . . . .	20
3.1	Objects and lighting conditions in seal pup images . . . . .	33
3.2	Isolated pixel removal and line dissection . . . . .	50
3.3	Segmentation results on sample images . . . . .	62
3.4	Performance of segmentation algorithm on training data . . . . .	65
4.1	One-class classification categories . . . . .	81
4.2	Final classification results . . . . .	108
4.3	Total error rate for optimal system . . . . .	111
B.1	Examples of segmentation performance evaluated as “excellent” . . .	135
B.2	Examples of segmentation performance evaluated as “good” . . . . .	137
B.3	Examples of segmentation performance evaluated as “satisfactory” . .	138
B.4	Examples of segmentation performance evaluated a “poor” . . . . .	139
B.5	Examples of segmentation performance evaluated as “failure” . . . . .	141

# List of Figures

2.1	Adult male harp seal . . . . .	7
2.2	Harp seal pup . . . . .	8
2.3	Harp seal whelping concentrations . . . . .	9
2.4	Survey photograph . . . . .	12
2.5	Simplified pattern recognition system . . . . .	22
3.1	Pre-processing of large images . . . . .	29
3.2	Complete software system . . . . .	30
3.3	MATLAB GUI used for cropping subimages . . . . .	31
3.4	Contrast stretching . . . . .	36
3.5	Image enhancement example . . . . .	37
3.6	Subdivision of images for adaptive thresholding . . . . .	39
3.7	Histogram distributions . . . . .	42
3.8	Bimodality example . . . . .	44
3.9	Skewness examples . . . . .	45
3.10	Overlap area in subdivided regions . . . . .	47
3.11	Adaptive thresholding results . . . . .	48
3.12	Canny edge detection . . . . .	49
3.13	Bitwise AND of negative Canny and threshold images . . . . .	50

3.14	“Cleaning” results . . . . .	51
3.15	Connected components . . . . .	52
3.16	Incomplete edge detection . . . . .	52
3.17	Distribution of area values for seal pup blobs . . . . .	53
3.18	First step of ICC algorithm . . . . .	54
3.19	Second step of ICC algorithm . . . . .	56
3.20	Pixel relabeling using 8-connected rule . . . . .	57
3.21	Example 1 of reattaching perimeter pixels . . . . .	58
3.22	Example 2 of reattaching perimeter pixels . . . . .	59
3.23	Final result of ICC algorithm . . . . .	60
3.24	Final result of segmentation algorithm . . . . .	61
4.1	A simple two-class classifier . . . . .	72
4.2	One-class classifier . . . . .	73
4.3	Overfitting . . . . .	76
4.4	Bias-variance trade-off . . . . .	78
4.5	Regions in one-class classification . . . . .	82
4.6	ROC example . . . . .	86
4.7	Graphical representation of a SVDD . . . . .	92
4.8	Chernoff distance . . . . .	101
4.9	Optimize kernel whitening on Parzen using AUC . . . . .	102
4.10	Optimize kernel whitening on SVDD using AUC . . . . .	103
4.11	ROC curve for Parzen classifier . . . . .	106

# List of Algorithms

1	Isolate Connected Components . . . . .	134
---	--	-----



# List of Abbreviations and Symbols

AUC	Area Under the ROC Curve
DFO	Canadian Department of Fisheries and Oceans
GPS	Global Positioning System
GUI	Graphical User Interface
ICC	Isolate Connected Components
MP	Megapixels
MSE	Mean squared error
NN	Nearest Neighbor
PCA	Principal Component Analysis
RGB	Red, green, and blue bands in a colored image
ROC	Receiver Operating Characteristic
ROI	Region of interest
SVDD	Support Vector Data Description
SVM	Support Vector Machine
TIFF	Tagged image file format
X mag.	Times magnification
cm	Centimeters
dpi	Dots per inch
mm	Millimeters
<i>a</i>	Area of a segmented object

$\mathbf{a}$	Center of the hypersphere in the SVDD
$b(t)$	Normalized between-class variance
$b_{\max}$	Bimodality coefficient
$C$	Regularization parameter in the SVDD which regulates the tradeoff between complexity and errors
$d$	Dimensionality of input patterns
$d'$	Dimensionality of principal components
$d(\mathbf{z} \omega_T)$	Distance of an object $\mathbf{z}$ to the target class $\omega_T$
$\mathcal{E}_{\text{true}}(f, \mathbf{w}, \mathcal{X})$	Total error or cost function for function $f$ with weights $\mathbf{w}$ over the complete feature space
$\mathcal{E}_{\text{emp}}(f, \mathbf{w}, \mathcal{X}^{tr})$	Empirical error or cost function for function $f$ with weights $\mathbf{w}$ over the complete training set
$\mathcal{E}_{\text{tot}}(f, \mathbf{w}, \mathcal{X}^{tr})$	Total error or cost function for function $f$ with weights $\mathbf{w}$ over the complete training set
$\mathcal{E}_{\text{struct}}(f, \mathbf{w})$	Structural error for function $f(\mathbf{x}; \mathbf{w})$
$\mathcal{E}_{\text{I}}(\mathcal{E}_{\text{II}})$	Cost for an error of the first kind (second kind)
$\mathcal{E}_{\text{M}}$	Error measurement computed by integrating $\mathcal{E}_{\text{II}}$ over all possible $\mathcal{E}_{\text{I}}$ , also known as the AUC
$f(\mathbf{x}), f(\mathbf{x}; \mathbf{w})$	Output of classifier with free parameters $\mathbf{w}$ given input $\mathbf{x}$
$f_{T+}$	Fraction of the target objects which are accepted by a one-class classifier
$f_{T-}$	Fraction of the target objects which are rejected by a one-class classifier
$f_{O+}$	Fraction of the outlier objects which are accepted by a one-class classifier

$f_{O-}$	Fraction of the outlier objects which are rejected by a one-class classifier
$h$	Kernel width for Parzen density estimation
$I(\cdot)$	Indicator function
$I(x, y)$	2D function representing an image where $(x, y)$ are image coordinates
$I_{in}$	Input image
$I_{out}$	Output image
$I_T(x, y)$	Threshold image
$\mathcal{I}$	Identity matrix
$J_c$	Chernoff distance between the target and outlier data
$K(\mathbf{x}_i, \mathbf{x}_j)$	Kernel function operation on objects $\mathbf{x}_i$ and $\mathbf{x}_j$
$l$	Length of a segmented object
$L$	Lagrangian, the combination of an error function and constraints; or number of intensity levels in image
$m_i$	Mean of class $i$
$M$	Total number of image pixels
$n$	Number of pixels in a given region
$n_i$	Frequency of the intensity level $i$
$N$	Number of objects in a training set
$p$	Degree of a polynomial kernel; also perimeter length of a segmented object
$p(\cdot)$	Probability density function
$p(\cdot \cdot)$	Conditional probability density
$p(\mathbf{z} \omega_T)$	Resemblance of an object $\mathbf{z}$ to the target class $\omega_T$

$p_{\mathcal{N}}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$	Normal or Gaussian distribution, characterized by mean $\boldsymbol{\mu}$ and covariance matrix $\Sigma$
$p_p(\mathbf{x})$	Parzen density estimation
$q, r$	Gray level/intensity of an image pixel
$r_{\min}$	Minimum gray level in image
$r_{\max}$	Maximum gray level in image
$R$	Radius of the hypersphere in the SVDD
$s$	Width of a Gaussian kernel
$t, T$	Image threshold
$t^*$	Threshold which maximizes between-class variance
$T(r)$	Transformation function used for contrast stretching
$w$	Width of a segmented object
$\mathbf{w}$	Parameter or weight vector
$\mathbf{w}^*$	Optimal parameters
$\mathbf{x}$	Column vector, most often representing an object
$x_i$	$i^{th}$ feature
$(\mathbf{x})^T$	Transpose of $\mathbf{x}$
$\mathcal{X}$	Feature space containing all possible objects $\mathbf{x}$
$\mathcal{X}_T$	Area in feature space which contains all target objects
$\mathcal{X}^{tr}$	Training set containing $N$ objects $\mathbf{x}_i$ with their labels $y_i$ ; in one-class problems often just target objects are present
$y_i$	Output label for pattern $\mathbf{x}_i$
$\mathbf{z}$	Column vector, most often representing a test object
$\mathcal{Z}$	Area in feature space which contains all outlier objects
$\alpha_i, \gamma_i$	Lagrange multipliers used to construct the SVDD

$\gamma$	Skewness
$\varepsilon(f(\mathbf{x}), y)$	Error or cost contribution of function $f$ by one training object $(\mathbf{x}, y)$
$\eta_{pq}$	Normalized central moment
$\theta$	Threshold on a probability or distance
$\lambda$	Regularization parameter (tradeoff parameter between the empirical and structural error)
$\mu$	Mean
$\mu_3$	Third moment about the mean
$\boldsymbol{\mu}$	Mean vector of a data set
$\xi_i$	Slack variables introduced in the SVDD to account for errors
$\sigma$	Standard deviation
$\sigma_i^2$	Variance of class $i$
$\sigma_T^2, \sigma_{total}^2$	Total variance
$\sigma_B^2$	Between-class variance
$\sigma_W^2$	Within-class variance
$\Sigma$	Full covariance matrix
$\Sigma_T, \Sigma_O$	Covariance matrix of the target, outlier class
$\phi_i$	Moment invariant $i$
$\Phi(x)$	Mapping of vector $\mathbf{x}$ to a high-dimensional kernel space
$\omega_T, \omega_O$	Target, outlier class
$\forall$	For all elements in a set
$\ \cdot\ $	2-norm (Euclidean distance)

# Chapter 1

## Introduction

### 1.1 Problem Statement

The Harp seal is an abundant, medium sized seal which lives in the North Atlantic. The largest harp seal population is the Northwest Atlantic stock. The Canadian and Greenland hunt for the Northwest Atlantic harp seal is the largest marine mammal harvest in the world. Therefore, it is important to monitor abundance and population trends to ensure that these removals are sustainable [1]. Harp seals are among the most important pinniped predators in the Gulf of St. Lawrence due to their abundance in this area. Large quantities of prey, such as Atlantic cod, are consumed by harp seals and other marine mammals impacting the yield of commercial fisheries. Therefore, information on population size of harp seals is also important in evaluating the magnitude of prey consumption [2]. The Department of Fisheries and Oceans (DFO) is responsible for managing the seal hunt and estimating population size. The total population model currently used by DFO incorporates estimates of pup production, age-specific reproductive rates, and age-structured removals [3].

Since 1990, visual and photographic aerial surveys have been flown over whelping

concentrations to determine pup production of Northwest Atlantic harp seals at four to five year intervals. Photographic surveys use a large format metric mapping camera to take black-and-white photographs of whelping areas. To count seal pups, manual analysis of aerial photographs is performed by trained scientific personnel who have extensive knowledge of the harp seal species. This process is very labor-intensive, expensive, and potentially error-prone. According to DFO, manually counting animals as part of population assessment is one of the most complex and time consuming tasks undertaken by the department [4]. An automated approach to counting seal pups in aerial photographs will reduce the amount of time required to compute population estimates and potentially improve the accuracy of pup counts.

## 1.2 Research Objectives

The main objective of this research is to develop an original automated method to recognize seal pups in black-and-white aerial photographs collected by DFO. Such a method should conserve time, reduce sources of error, and save money for estimating total pup counts. To accomplish this objective, image segmentation and pattern recognition algorithms for the detection and classification of harp seal pups are studied and implemented.

This research must also meet Computational Science<sup>1</sup> objectives. Computational Science studies the use of computers in analyzing, interpreting and solving complex scientific problems arising in natural sciences (chemistry, physics, earth sciences and mathematics) and engineering. Objectives for this program include training students in:

---

<sup>1</sup>A detailed description of the Computational Science Graduate Program at Memorial University of Newfoundland can be found online at <http://www.mun.ca/science/CMSC/index.php>.

1. state-of-the-art numerical methods;
2. high performance computing;
3. use of graphics, visualization, and multi-media tools;
4. the acquisition, processing, and analysis of large experimental data sets; and
5. applying these techniques to at least one scientific or engineering discipline.

The problem addressed in this research is engineering-based and encompasses all of these objectives: advanced linear algebra methods are applied in several techniques used in this thesis including Principal Component Analysis, kernel whitening, and many image processing algorithms; large data sets of target objects (seal pups) are acquired, processed, and analyzed by segmentation and classification algorithms; graphics and visualization is inherent in implementing and explaining the image processing techniques used; and all algorithms are implemented in MATLAB[5], a programming environment designed for the very purpose of solving computational problems.

## 1.3 Thesis Contributions

The following contributions have been made to the fields of image processing, pattern recognition, and environmental monitoring through this research:

1. A novel algorithm is designed to address the automated counting of seal pups from digitized aerial photographs. This process uses a combination of image segmentation techniques and classification methods.



2. A shape-preserving image segmentation algorithm has been developed to isolate seal pups from highly cluttered background. The existence of shadows and the large variation of pixel values over seal objects pose unique challenges. To isolate regions of interest, this new approach first utilizes contrast stretching and adaptive thresholding using between-class variance and histogram skewness. A “cleaning” algorithm is then applied to the threshold image using edge detection, line dissection, and removal of objects based on size constraints. Finally, a unique algorithm called Isolate Connected Components further isolates target objects using object labeling and removal/growing of perimeter pixels.
3. Instead of applying a generic classifier, a relatively new classification technique called one-class classification is used to robustly identify target objects. Two different one-class methods have been applied for classification of seal pups, namely the Parzen density estimation and the Support Vector Data Description.
4. Initial recommendations have been made to DFO for developing a fully automated software tool for population estimation of harp seal pups.

## 1.4 Thesis Organization

This document is organized into five chapters. Chapter 2 describes background information on the harp seal species and examines the manual process currently used to count seal pups in aerial images. A literature review on computer-aided systems used in ecological science and environmental monitoring is conducted with a focus on automated population assessment research. This is followed by a description of the general approach used in this research to develop a computer-aided system for recognition of seal pups in aerial images. Chapter 3 describes the image segmentation

algorithm used to extract potential seal pup objects from aerial images and defines the features used to represent these objects. Chapter 4 focuses on the development of a classifier for harp seal pups. An overview of one-class classification theory is given, followed by a description of the one-class classifiers applied to the problem at hand, the Parzen density estimation and the Support Vector Data Description. Classifier optimization and feature reduction techniques including Principal Component Analysis and kernel whitening are reviewed. Results are presented including false positive and false negative rates on each classifier tested. Chapter 5 summarizes the results and provides recommendations for future work. Throughout the thesis, a certain degree of familiarity with introductory statistics, image processing and classification is assumed. Non-trivial concepts are explained within each chapter where necessary.

# Chapter 2

## Background

This chapter presents background information on the harp seal species. A description of recent methods for assessing the Northwest Atlantic harp seal population provides motivation for this current work. This is followed by a summary of computer-aided systems used in ecological science and environmental monitoring, in particular, automated population assessment of wildlife species. Finally, the automated approach for recognizing harp seal pups in aerial images developed in this research is presented.

### 2.1 Harp Seals

Harp seals, *Pagophilus groenlandicus*, are the most abundant pinniped (fin-footed animal) in the Northwest Atlantic [1, 6]. They owe their name to the irregular horseshoe-shaped band of black straddling the back in the adult male (see Figure 2.1). This band, or “harp”, unites across the shoulders, curves down toward the abdominal region and then back toward the posterior flippers where it abruptly disappears. The background color of the pelt is steel blue when wet and pale gray when dry. The head and tail are black, while the anterior flippers and belly are whitish.

Adult females are similarly patterned, except that the harp, the head, and the tail are usually somewhat lighter in colour. Some adult females have irregular dark gray spots on the back with no clearly defined harp [7]. Male harp seals are only slightly larger than females with adults averaging 1.6 meters in length and 130 kilograms in weight [8].



Figure 2.1: Adult male harp seal. From DFO marine mammal section.

Harp seals migrate annually between Arctic and sub-Arctic regions of the North Atlantic Ocean. They are confined to three widely separated populations breeding in the White Sea north of Russia, the “West Ice” near Jan Mayan Island southeast of Spitsbergen, Norway, and off Newfoundland. The Northwest Atlantic harp seal population, historically the largest, summers in the Canadian Arctic and Greenland. In late September when new Arctic ice is forming, the seals start their journey south along the east and west coasts of Baffin Island and eastward through Hudson Strait. The migrating group of seals separate into two herds; one breeding on the southward drifting Arctic pack ice off Southern Labrador and Northern Newfoundland (called the “Front” sub-population) and the other breeding on ice in the Gulf of St. Lawrence near the Magdalen Islands (called the “Gulf” sub-population). During January and February seals disperse widely and feed intensively. Feeding is particularly important for pregnant females, for they need energy to support the enormous demands of their rapidly growing offspring during lactation [7].

Pregnant females give birth, or *whelp*, several days after they have hauled out onto

the winter pack ice in late February or early March. Females nurse a single pup (twins are uncommon) for approximately twelve days and then mate and disperse. Newborn pups are about 85 centimeters long, weigh about 11 kilograms and are yellowish in colour. In about 3 days their fur turns to a fluffy white from which the pups derive the name *whitecoat* (see Figure 2.2). The whitecoat moults its white fur at about three weeks of age. Large moulting concentrations, called *patches*, form on the sea ice off Northeastern Newfoundland and in the Northern Gulf of St. Lawrence in April and May (see Figure 2.3). After the moult, the seal pups disperse and eventually migrate north again [8].



Figure 2.2: Harp seal pup, also called a *whitecoat*. From DFO marine mammal section.

### 2.1.1 Population Assessment

Harp seals are harvested commercially and for subsistence purposes in Atlantic and Arctic Canada and from waters around Greenland. Harps are also taken as by-catch in commercial fisheries. Commercial harvesting of the species dates back to the early eighteenth century. The Canadian and Greenland hunt for Northwest Atlantic harp seals is the largest marine mammal harvest in the world. To ensure a sustainable yield, it is important to monitor abundance and population trends on a regular basis. In Canada, the commercial harvest is limited through a management plan that outlines management objectives, catch levels, methods of hunting, and seasonal and regional



Figure 2.3: Map of four whelping concentrations located in the Gulf of St. Lawrence and off Newfoundland and Labrador during March 2004. Shading indicates areas covered by reconnaissance surveys conducted by DFO. The general direction of drift is indicated by the arrows. From DFO, 2004 Harp Seal Survey [6].

closures [1, 9]. The objective of this plan is to ensure that the population does not decline below a precautionary reference level.

Marine mammals, because of their large size and abundance, may have an important influence on the structure and function of many marine ecosystems [10]. One obvious impact is that large quantities of prey are consumed by marine mammals, to the detriment of commercial fisheries. Therefore, in addition to species management, assessing harp seal population is important in estimating the consumption of Atlantic cod, Atlantic herring, and other prey by the seals. Estimates of prey consumption by harp and gray seals were developed by modeling changes in population size, energy requirements, diet composition, and changes in population distribution [2].

The total population of harp seals cannot be counted directly. During the summer, surveys of the total population are impractical because harp seals are distributed widely across the Arctic and North Atlantic. Seals congregate during whelping and moulting periods, but not all of the population is present on the surface at any one time and place. However, whitecoats remain on the ice while being nursed. Therefore, seal populations can be assessed by estimating pup production as a first step. This is combined with information on pregnancy rates and age-structured removals to construct a total population estimate [3, 8].

Prior to 1990, the annual pup production of harp seals was estimated using a variety of techniques including survival indices, catch-at-age analysis, sequential population models [11, 12, 13, 14, 15], aerial photographic surveys [16, 17] and mark-recapture experiments [18, 19]. Unfortunately, these different techniques often produced conflicting estimates. Since 1990, pup production has been estimated using a combination of photographic and visual aerial surveys [6, 20, 21]. By consistently using these methods, comparable estimates can be used to determine if pup production has increased in recent years.

### **2.1.2 Aerial Visual and Photographic Surveys**

Whelping concentrations are identified by conducting fixed-wing and helicopter reconnaissance surveys of areas historically used by harp seals. Once located, abundance estimates of seal pups are determined using visual and photographic methods.

Visual surveys are flown via helicopter over the whelping concentrations. A series of equally-spaced parallel lines, called transects, are laid out prior to the flight. As the transects are flown, two observers seated in the rear of the helicopter count all pups within a pre-defined visual area on each side of the aircraft. Correct altitude and

transect spacing are maintained using a radar altimeter and GPS navigation systems [6]. Such surveys are not always practical since they can only be conducted in good weather conditions and with the assistance of a support vessel.

Photographic surveys are advantageous over the visual method since they allow personnel to scan for seals in a laboratory setting after the survey is complete. A fixed-wing aircraft, equipped with a large format metric mapping camera with motion compensation, is used to take black-and-white photographs of whelping areas (Figure 2.4). Surveys are conducted at a fixed altitude. Transect lines and spacing for all surveys are mapped prior to the flights to ensure complete coverage of the patches. Any overlap between photos within a transect is removed prior to analysis. Correct altitude and transect spacing is maintained using barometric altimeters and GPS navigation systems. Ice drift is monitored by satellite transmitters to ensure that transects remain independent [20]. Although the general method for conducting photographic surveys has been consistent since 1990, physical parameters such as altitude of surveys and location of transect lines over whelping concentrations may vary. A description of the physical parameters, camera equipment, and film used to capture images for this research is given in Section 3.2.1.

### **2.1.3 Manual Photograph Analysis**

Analysis of aerial photographs is performed by *readers*, trained scientific personnel with extensive knowledge of harp seals and the environment. Photographs are examined by several readers using an illuminated hand-lens (7-8X mag.) or a rail-mounted low magnification binocular microscope. To standardize the readers prior to the actual readings, each examines a common series of photographs and compares identified seals. Once the cues used to identify seals are consistent among readers, all photos



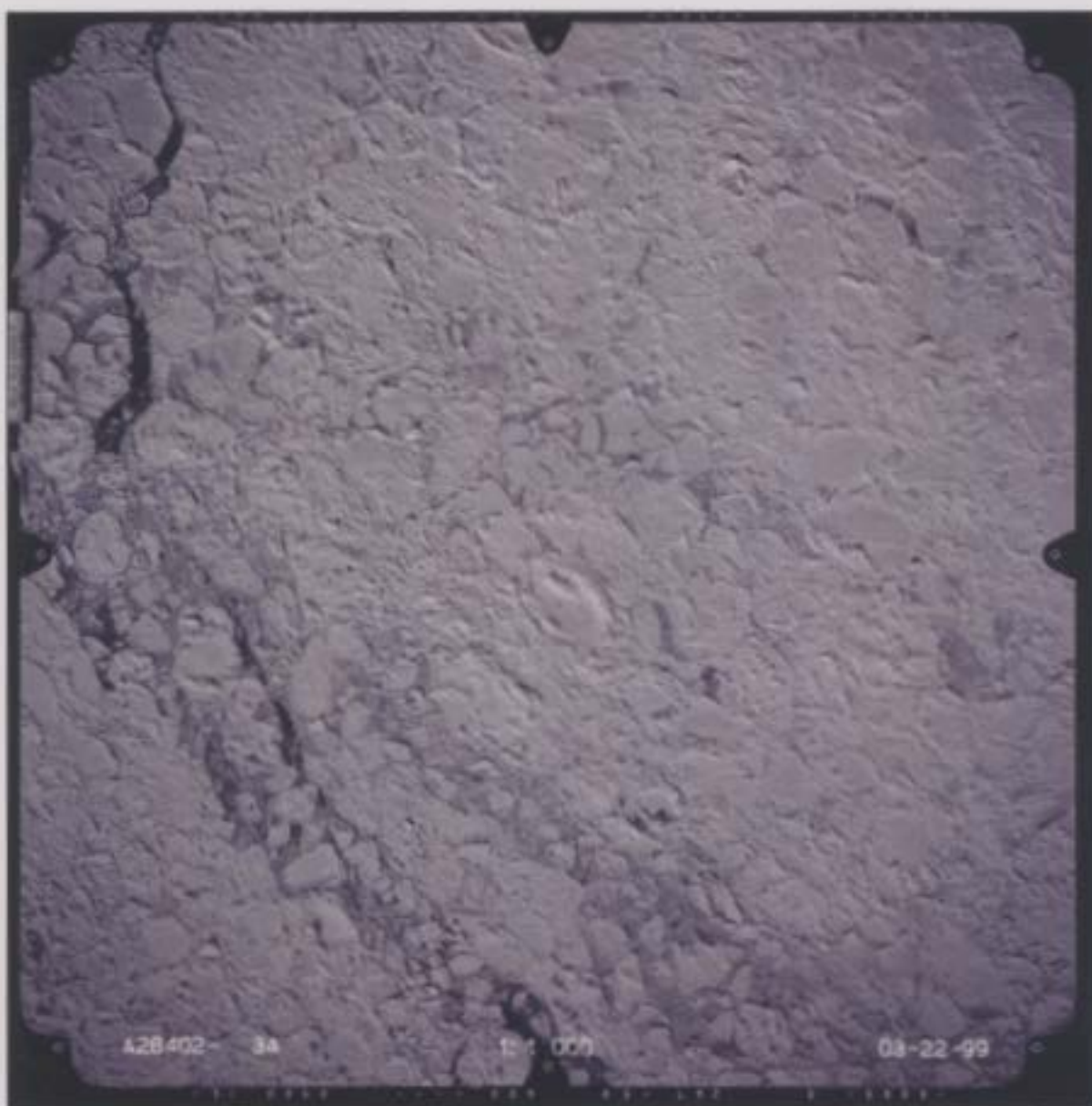


Figure 2.4: Survey photo of whelping patch along the Front, March 1999. This image covers an area of 274.3 meters squared.

are read once by each reader. For each photograph, the number and position of all pups are recorded on a clear acetate overlay.

Manual identification of harp seal pups is not always conclusive, even for the very experienced reader. Therefore, measures are taken to ensure the most accurate final reading. After all photographs are examined, each reader re-reads a series of their photographs in sequence to determine if identification of seals has improved over the course of the readings. Readings of photographs continue until the counts from the first and second reading differ by less than 5%. If pup counts differ by more than 5%, the counts from the first reading are replaced by those from the second reading [6].

To correct for misidentified pups, a series of randomly selected frames for each patch are examined by all readers. All resulting acetates are then overlaid and re-examined by a few experienced readers to determine a ‘best estimate’ of the number of pups present. Any pup that cannot be positively identified is not included. The original counts are regressed on the best estimate to determine a correction factor for each survey and reader. The corrected counts for each photograph are then summed to obtain the corrected count for each transect. The variance associated with the reading corrections is summed over transects to estimate the total measurement-error for the survey and added to the sampling variance. A correction for the temporal distribution of births and loss of pups, due to the dispersion of ice packs or pups in the water, may also be applied to total pup estimates. For a review of the mathematical equations used for estimating total pup production and error variance, please refer to Stenson *et al.* [20, 22].

Manual analysis of aerial photographs can take many months and involve several people. Each survey can take up to 3 person-years to analyze and manpower costs are quite high. The purpose of this work is to initiate an automated process to perform the activities described above in order to improve accuracy and reduce time and costs.

## 2.2 Previous Work

In recent years, pattern recognition and computer-aided systems have become an important tool in ecological science and environmental monitoring. Such systems are being employed for species identification, population counting, and determining the size of organisms. It has been shown that automating these tasks produce faster and more accurate results when compared to manual implementation.

The practice of species identification is one area where computer-based systems is a growing trend. Worldwide, there is an increasing need for biodiversity monitoring, while at the same time the number of trained taxonomists declines. Demand for routine identification far outstrips the capabilities of the taxonomic community [23]. Systems have been developed for automated species identification through hair patterns [24], genomic data [25], sound [26, 27, 28] and image analysis [29-37]. The latter has been used to identify zooplankton [29], moths [30], spiders [31, 32], stonefly larvae [33, 34], and other insects [35, 36, 37]. Image-based species classification systems are often trained on images of dead specimens in controlled lighting conditions. Live specimens in their native habitat may move when an image is being captured and lighting conditions are typically unpredictable. The demand for computer-based systems that can automatically identify the species of live plants, insects, or animals from digital images or recordings is surely to increase in the future [30].

Pattern recognition and computer-aided systems have also been developed for determining the size of organisms and counting populations. Sonar systems have been employed for automated fish sizing and counting [38, 39]; thermography and multi-spectral scanners have been tested for detecting and counting deer [40, 41] and geese [42, 43]; and image analysis programs have been designed for determining population size of several species including plankton [29, 44], stonefly larvae [34], waterfowl

[45, 46, 47, 48], penguins [49], whales [50], sea-lions [51], and caribou [48].

One of the most common image analysis methods for estimating population size of wildlife is counting animals in aerial photographs. Traditionally, this approach has been conducted manually. For example, manual analysis of aerial images has been used to count arctic seabirds [52], Adelie penguins [53], flamingoes [54], caribou [55], elk [56, 57], beluga whales [58], and seals [59, 20, 6]. However, manual counts from aerial photographs are labor-intensive and can be subject to considerable error. While computer-aided wildlife census may save time and improve accuracy of counts, limited research has been conducted in this area [48]. The following paragraphs summarize recent efforts in automating population counts from aerial images.

In 1988, Gilmer *et al.* [45] conducted aerial photographic surveys of snow geese and Ross geese with a hand-held camera and high resolution film. After processing, negatives were masked to eliminate areas without geese and then enlarged onto photo paper. Photos were digitized using a linear array image-scanner and read into a computer for analysis. The number of geese within two small polygons (training sites), chosen at random, provided the training statistics necessary for determining threshold values and goose-per-pixel relationships. Threshold values partitioned the image into two classes, 'goose' and 'non-goose' (water, soil, vegetation). The threshold value was chosen iteratively by visually comparing the density-sliced, digital images on the display monitor with the original aerial photos. The area of an image (in pixels) classified as 'goose' was divided by the average area of a single goose (computed manually using the training sites) to estimate the total number of geese on the photo. This early attempt at automatically counting geese in aerial photos provided fairly accurate counts of white geese at significant savings of time and effort. However, results may vary if background habitats contain density values corresponding to those of the images being quantified, and will depend on the size distribution and spatial

orientation of birds on the photograph.

In 1990, Bajzak and Piatt [46] presented a semi-automated technique for computer-aided counting of snow geese from aerial images. Besides counting birds, this method can be used to sort birds into size and tonal (photographic density) classes. A transparency is produced from the aerial film and then digitized using a scanning microdensitometer. The digitized image is analyzed using two computer programs. The first program produces a printed output of density values from a specified sub-image which is used to manually determine the required parameters for computer identification and counting. These parameters include tonal range of snow geese and the minimum and maximum number of pixels that represent each bird. After performing experiments to fine-tune these parameters, the second program identifies individual birds based on these parameter values, counts the number of pixels per bird, and calculates the minimum, maximum, and average densities for each identified bird. Final results showed only a 2.3% difference between visual and computer counts, an improvement from Gilmer *et al.* [45]. The total time required to analyze an image was reduced even though preliminary data analysis and establishing relevant parameters is time consuming.

A more automated approach was developed by Gosine et al. [51] in 1995 to count sea-lions in aerial images, video, or still pictures. A binary image is first produced by thresholding an enhanced edge image. Objects are manually identified by the user as ‘sea lion’ and ‘not-sea lion’ for the first image frame in the video. For each object, basic features (e.g. size, shape, mean, standard deviation) and a specialized intensity gradient across the object are computed and stored in a database. This information is used to train a nearest-neighbor classifier [60] that is applied to subsequent video frames to discriminate ‘sea lions’ from ‘not-sea lions’. The results indicated good agreement between manual and automated counts.

In 1996, Cunningham *et al.* [47] adapted MacIntosh-based, public-domain software to create DUCK HUNT, a semi-automated program for counting waterfowl. The software accesses digital images and, if necessary, performs interactive enhancements using spatial and spectral processing. Objects to be counted are selected by their spectral reflectance using interactive density slicing of images. When all objects of interest are highlighted, the mouse cursor is used to mark a sample of highlighted objects to define size and shape parameters for objects of interest. A counting routine then counts objects with similar features. The selection parameters used for counting can be saved and used for processing other images with similar optical characteristics. This technique showed promising results, but is semi-automated. Efficient application of this program depends on high contrast between objects of interest and background, large concentrations of objects of interest, separation of individuals, and consistent image quality.

In 2003, Laliberte and Ripple [48] used public-domain image-analysis software, ERDAS Imagine[61] and ImageTool[62], to assess accuracy of counting wildlife from remotely sensed images. Their objective was to develop a method that was simple enough to permit widespread use, requiring only basic knowledge of image processing techniques. Four illustrative case studies were chosen: a black-and-white aerial photo of snow geese on water using a mapping camera; a color aerial photo of Canada geese using a hand-held camera; a black-and-white aerial photo of caribou using a mapping camera; and a high-resolution satellite image (IKONOS) of cattle. The general approach involved windowing out smaller sub-images, applying filters to enhance the images, separating animals from the background by manually thresholding based on sub-image histograms, and then using spectral and area attributes to separate single animals from groups for counting purposes. Using manual counts for comparison, computer count errors were computed to be 2.8% for the snow geese image, 4.4% for

the Canada geese image and 10.2% for the caribou image. The test with the satellite image performed satisfactorily and showed promise for future applications, however ground-truth data was not available to compute error rates.

In 2004, Trathan [49] derived population estimates of Macaroni penguins using computer-based image analysis of color aerial photographs digitized with a photogrammetric scanner. Using MATLAB, images are divided by defining a separate polygon around each colony of penguins. Each colony image is segmented into a grid for further analysis. Within each grid square, or region-of-interest (ROI), descriptive statistics are used to select the color band (red,green,blue) that best discriminates between penguins and their background. A random line transect covering several penguins is drawn through the ROI. A smaller random area within the ROI is selected and a histogram of pixel values computed. Pixel values along the transect line and from the pixel histogram are used to compute a threshold value for the image. After the threshold is applied, pixel values classified as penguins are smoothed with a median filter and a second threshold is applied to eliminate any remaining background pixels. The pixel-area of resulting blobs is used to remove objects not characteristic of penguins. Each separate ROI is reassembled into a single image and an automated routine is applied to count the number of birds present within the colony. Results were highly correlated with manual photograph counts. However, many of the steps described above are only semi-automated (e.g. defining ROIs, inspection of histograms, selecting a threshold) and therefore the process is time consuming.

In 2006, Mills [50] developed an automated image analysis system, called Marine Mammal Detector, to detect and classify beluga whales in digitized aerial photographs. In general, a filtering algorithm masks image pixels that are considered “unreadable” (land, sun glare, extensive wave crests, image borders). A specialized adaptive thresholding technique is used to segment potential whales and then a size

filter is applied to remove objects that are obviously not whales. Segmentation is improved by applying a watershed algorithm and further thresholding to separate adjacent whales. A Support Vector Machine [63] classifier is used to classify objects as either ‘whale’ or ‘not-whale’. To optimize the classifier, a genetic algorithm is applied for feature reduction and classifier parameter calibration during training. Testing of this method demonstrated an excellent separation of classes and a low false positive rate.

In 2004, DFO and Memorial University of Newfoundland initiated research in the development of an automated system to identify harp seal pups in aerial photographs. In an initial attempt to tackle this problem, Hogan *et al.*[64] used Matrox Inspector 4.1[65] and MATLAB to perform image processing and analysis methods to segment objects characteristic of seal pups. A combination of image filtering, seeded region growing and morphological operations were employed to segment target objects. Error rates on this method were not available. An exhaustive search could not find evidence of any other automated system to identify harp seal pups in aerial photographs. The research conducted here is the first attempt of a more complete approach that includes segmentation, feature analysis and classification of seal pups.

Table 2.1 gives a comparative summary of the aforementioned efforts in automated population detection and counting in aerial images. The first two columns list the author and year of the research and the species being studied. The third column specifies whether the segmentation algorithm is fully automated, manual (requires human interaction for each step), or a combination of the two (semi-automated). The fourth column specifies if a classification algorithm is applied and the final column gives a brief description of the segmentation and classification methods (if applicable).



Table 2.1: Summary of recent efforts in automated population detection and counting from aerial images.

Author, Year	Species	Segmentation	Classification	Methods Used
Gilmer, 1988 [45]	Geese	Semi-Automated	No	Training statistics (threshold values and goose-per-pixel relationships) acquired from training sub-images are used to partition image into 'goose' and 'non-goose' classes and then estimate total geese in photo.
Bajzak and Piatt, 1990 [46]	Geese	Semi-Automated	No	Manually selected parameters (tonal range, min and max pixel values) in specified sub-image are used to identify target species in remaining image(s).
Gosine, 1995 [51]	Sea-lions	Semi-Automated	Yes	Segmentation of ROIs using binary edge image produced by thresholding. Object features and specialized intensity gradient used to train a nearest-neighbor classifier.
Cunningham, 1996 [47]	Waterfowl	Semi-Automated	No	ROIs selected using spectral reflectance. Size and shape parameters of ROIs used to identify/count objects with similar features.
Laliberte and Ripple, 2003 [48]	Geese, caribou and cattle	Manual	No	ERDAS Imagine and Image-Tool are used to segment sub-images using filtering and thresholding and to analyze spectral and area attributes.
Trathan, 2004 [49]	Penguins	Semi-Automated	No	Segmentation using descriptive statistics, various thresholding techniques and median filtering.
Mills, 2006 [50]	Beluga whales	Automated	Yes	Segmentation using filtering, adaptive thresholding, and a watershed algorithm. Classification using an optimized SVM.
Hogan, 2005 [64]	Harp seal pups	Automated	No	Segmentation using filtering, region growing and morphological operations.

## 2.3 The Approach

Pattern recognition is the act of taking in raw data and making an action based on the “category” of the data [66]; it aims to classify patterns in data based on either *a priori* knowledge or on statistical information extracted from the patterns. Applications of pattern recognition now include: character recognition; target detection; medical diagnosis; biomedical signal and image analysis; remote sensing; identification of human faces and of fingerprints; reliability analyses; socioeconomics; archaeology; speech recognition and understanding; machine part recognition; automatic inspection; and many others [67]. Such systems are quite complex and can typically be partitioned into five main components:

1. **Sensing** - a sensor converts physical inputs, such as images or sounds, into signal data;
2. **Segmentation** - sensed objects are extracted from the signal data;
3. **Feature extraction** - object properties, or *features*, are measured;
4. **Classification** - a classifier uses features to assign the sensed object to a category, or *class*; and
5. **Post-processing** - a post-processor uses the output of the classifier to recommend actions.

Using these five components, a simplified system for the recognition of harp seal pups is presented in Figure 2.5. In the research presented here, steps 2 through 4 are addressed: segmentation, feature extraction, and classification. In the design of these components, a number of different activities are considered: collection of training

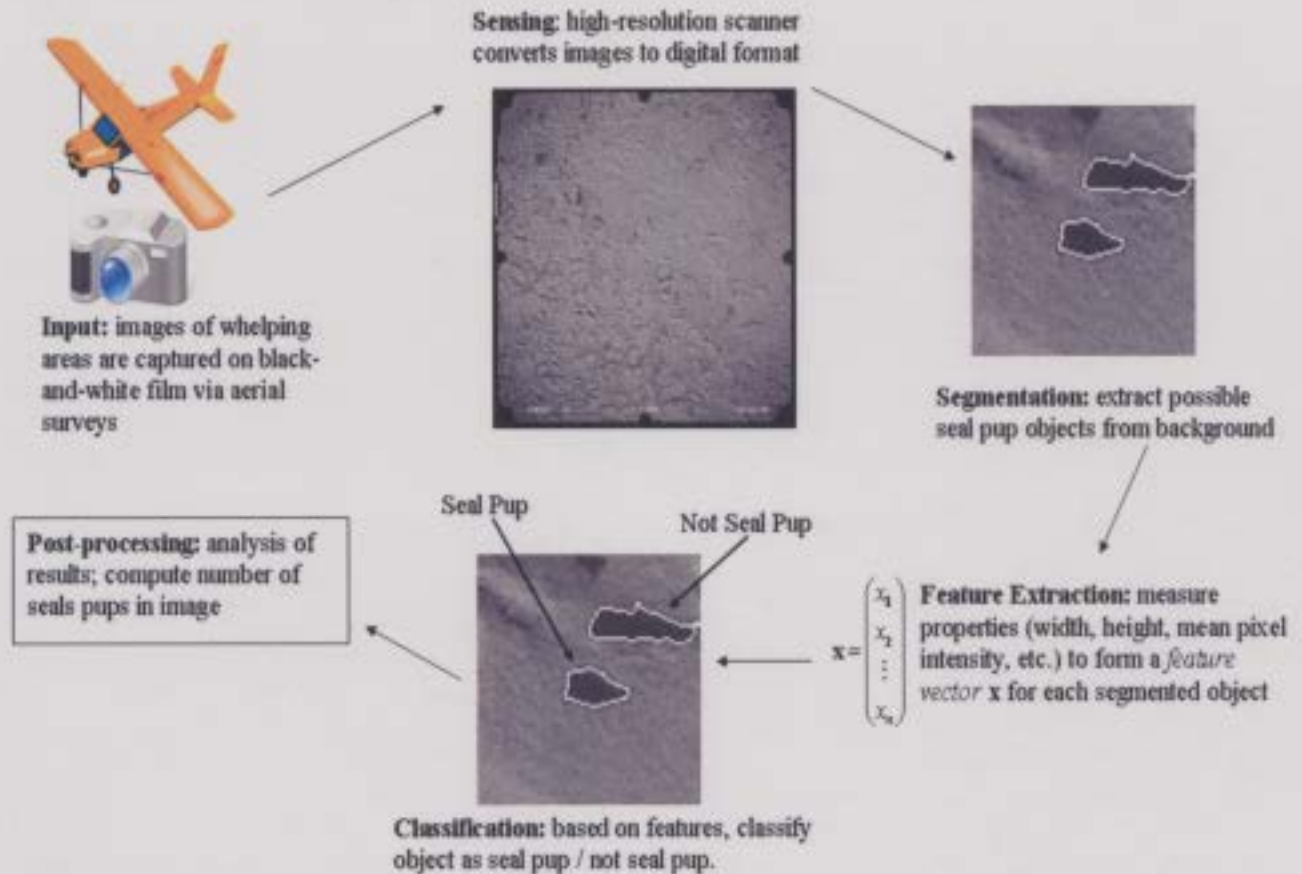


Figure 2.5: Simplified pattern recognition system for harp seal pups.

and test data, choosing distinguishing features, deciding on a classification method, training the classifier, and evaluating its performance on test data.

Training and test data are extracted from aerial images using a segmentation algorithm. Segmentation of nontrivial images is one of the most difficult tasks in image processing. It involves distinguishing between objects-of-interest and “the rest”, also referred to as the background. In segmenting images of whelping areas, the objects of interest are harp seal pups and everything else (ocean, ice, land, other animals, etc.) comprises the background. A combination of image processing techniques including contrast stretching, adaptive thresholding, edge detection, line dissection, and an original algorithm called Isolate Connected Components are used to segment seal

pups. Feature measurements are then computed for the extracted seal pup objects. A detailed description of the segmentation algorithm and object features is given in Chapter 3.

Due to a variety of objects and lighting conditions in input images, it is highly unlikely that the segmentation algorithm will work perfectly for every image (i.e. extract every seal pup and only seal pups). Therefore, the goal is to maximize the number of seal pups segmented while eliminating as much background as possible. If a background object is incorrectly segmented as foreground, it then becomes the responsibility of the classifier to identify it as a non-seal pup object.

A classifier is a function that takes a set of features that characterize an object and uses them to determine the type, or *class*, of each object. In many classification problems explicit rules do not exist to categorize an input object, but examples of objects from each defined class can be obtained. Therefore, a classifier can be constructed based on a finite set of pre-labeled training examples.

In a conventional multi-class classification problem, training examples are available for two or more classes. In a one-class classification problem, it is assumed that information for only *one* of the classes, the *target* class, is available; information about all other objects, or *outliers*, is minimal or not available. The objective is to define a boundary around the target class so that it accepts as many target objects as possible while minimizing the chance of accepting outlier objects.

For the current research, a one-class classification approach is very appropriate. The target class consists of segmented seal pups for which ample training data is available. While some outlier data is available (e.g. sea ice, ocean), it is unknown whether these samples are well-representative of all possible non-seal pup objects. Therefore, these objects are grouped into the outlier class. For training the classifier, only objects segmented from training data that are verified as seal pups will be used.

Verification is performed by comparing segmented objects with ground-truth images provided by DFO. For testing the classifier, all objects segmented from test data are input and then labeled as either ‘seal pup’ (target) or ‘not seal pup’ (outlier). It is important to note that the training and test data sets are mutually exclusive.

Many different models have been proposed for one-class classification and these are categorized into three approaches: density estimation, boundary methods, and reconstruction models. In this research, two one-class classifiers are considered: Parzen density estimation (density method) and the Support Vector Data Description (boundary method). A description of these classifiers and how they are optimized and applied in this research is given in Chapter 4.

# Chapter 3

## Image Segmentation

This chapter explains the segmentation algorithm used to extract target objects (seal pups) from the aerial photographs collected by DFO. First, a brief introduction to image processing and segmentation is given followed by a description of how the aerial photographs were captured, digitized, and reduced to manageable size. Next, a review of challenges presented by the complex images under study is given. This is followed by a detailed account of the segmentation algorithm which includes: contrast stretching; adaptive thresholding using between-class variance and histogram skewness; a “cleaning” algorithm that uses Canny edge detection, line dissection, and removal of objects based on size constraints; and an original procedure called Isolate Connected Components (ICC) that separates adjacent objects with minimal distortion to object shape. Segmentation results on training and test data are given. Finally, object features used by the classification component are described.

### 3.1 Introduction

An image may be defined as a two-dimensional function,  $I(x, y)$ , where  $x$  and  $y$  are spatial coordinates, and the amplitude of  $I$  at any pair of coordinates  $(x, y)$  is called the *intensity* or *gray level* of the image at that point. When  $x$ ,  $y$ , and the amplitude values of  $I$  are all finite, discrete quantities, the image is called a *digital image*. Each discrete  $(x, y)$  coordinate in a digital image is referred to as a *picture element* or *pixel* [68].

The field of *digital image processing* refers to processing images by means of a digital computer. For the purpose of digital processing, images are stored as two-dimensional arrays (matrices) in which each element of the matrix corresponds to a single pixel in the displayed image. Digital images require so much storage and computational power that progress in the field of digital image processing has been dependent on the development of digital computers and of supporting technologies that include data storage, display, and transmission. The first computers powerful enough to carry out meaningful image processing tasks appeared in the early 1960s. Since then, image processing techniques have been used in a wide variety of fields including computer vision, robotics, artificial intelligence, remote sensing, manufacturing, civil engineering, astronomy, geology, geophysics, biology, physiology, medicine, aerospace and defense, environmental monitoring, agriculture, marine sciences, crime and fingerprint analysis, movies and entertainment, and multimedia [69, 70].

When analyzing objects in images it is essential to distinguish between the objects of interest and “the rest”, also referred to as the background. The techniques used to find the objects of interest are typically referred to as *segmentation techniques* - segmenting the foreground from the background. In segmenting images of whelping areas, the objects of interest are harp seal pups and everything else (ocean, ice, land,

other animals, etc.) comprises the background. All subsequent interpretation tasks - feature extraction, object recognition, and classification - rely heavily on the quality of the segmentation process [71].

Segmentation of nontrivial images is one of the most difficult tasks in image processing. There is no universally applicable segmentation techniques that will work for all images. The choice of one segmentation technique over another is dictated mostly by the particular characteristics of the problem being considered. Image segmentation algorithms are generally based on one of two basic properties of intensity values: discontinuity and similarity [68]. In the first category, the approach is to partition an image based on abrupt changes in intensity, such as edges in an image. The principal approaches in the second category are based on partitioning an image into regions that are similar according to a set of pre-defined criteria. Thresholding, region growing, and region splitting and merging are examples of methods in this category. For a detailed overview of image processing and segmentation techniques, please refer to Woods and Gonzalez [68].

## **3.2 Data Set**

This section explains how the seal pup data was acquired, converted to digital format, and reduced in quantity to maximize computational efficiency. Features typically found in aerial images of whelping concentrations are discussed, along with challenges presented by these features.

### **3.2.1 Data Acquisition**

The harp seal data examined in this research was collected by DFO in 1999. Although the general methods for conducting visual and photographic surveys have



been consistent since 1990, physical parameters such as altitude of surveys and location of transect lines over whelping concentrations may vary. The methods and parameters described here were used to estimate harp seal pup production for 1999 and are explained in detail by Stenson *et al.* [20].

Fixed-wing photographic surveys were flown using two planes equipped with 23 x 23cm format metric mapping cameras (Zeiss RMK/A) with a motion compensation mechanism and Kodak Double-X (2405, ISO A4000) aerographic black-and-white film. The cameras were fitted with a 150mm Sonnar lens, and surveys were conducted at constant altitude of 183 meters. Each aerial photograph covers a geographical area of 274.3 meters squared. As explained in Section 2.1.3, trained readers analyze each developed photograph and record the position of all pups on a clear acetate overlay. Photographs with manually identified seal pups are known as *ground-truth* images and are used to select data for training and testing the segmentation and classification algorithms.

### 3.2.2 Data Digitization and Reduction

In order to implement an automated, computer-assisted approach to detecting and counting seal pups, it is necessary to convert aerial photographs to digital format. Aerial photographic film requires specialized equipment that is capable of scanning large format negatives at high resolution. Therefore, the images were scanned by a third party company that specializes in such tasks. Each scanned image was saved in TIFF (8-bit) format at a resolution of 907 dpi resulting in a total image size of 8430 x 8429 pixels (approximately 71 megapixels).

A significant amount of computer memory is required to store and perform operations on such large images. In order to maximize computational efficiency, the

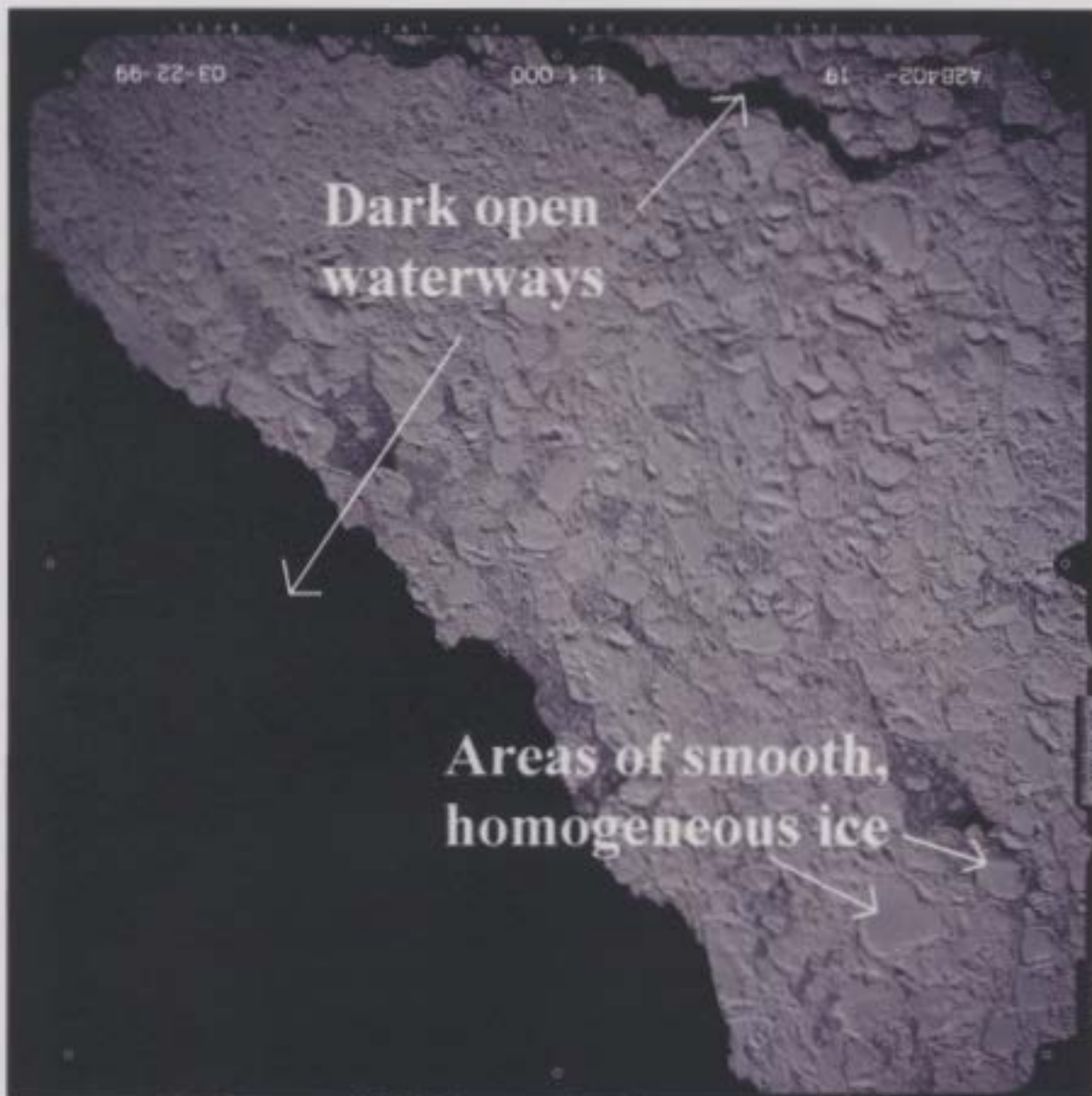


Figure 3.1: In a complete software system, a pre-processing step may exist in which regions void of seal pups, such as dark open waterways and vast areas of smooth ice with homogeneous gray levels, are masked out prior to cropping subimages.

71MP images are divided into more manageable subimages. In a complete software system for counting seal pups, this step would be automated; an original large image would be input to the system, analyzed for areas likely to contain seal pups, and these areas would be automatically divided into subimages for further processing. A pre-processing step may exist in which certain regions or features in the original image are masked out prior to creating subimages. Examples include dark open waterways that would camouflage seal pups and vast areas of smooth sea ice with homogeneous gray levels that clearly do not contain seals (see Figure 3.1). Masking out these

features before processing would reduce the complexity of the data set and improve computational time.

While image pre-processing and creating subimages are necessary components of a complete software system, this research focuses exclusively on segmentation, feature extraction, and classification; some automated pre- and post-processing steps, such as dividing the original image into subimages, have been excluded. The processing diagram for a complete software system is shown in Figure 3.2. The components contained within the dotted line have been implemented in this research.

### Complete System

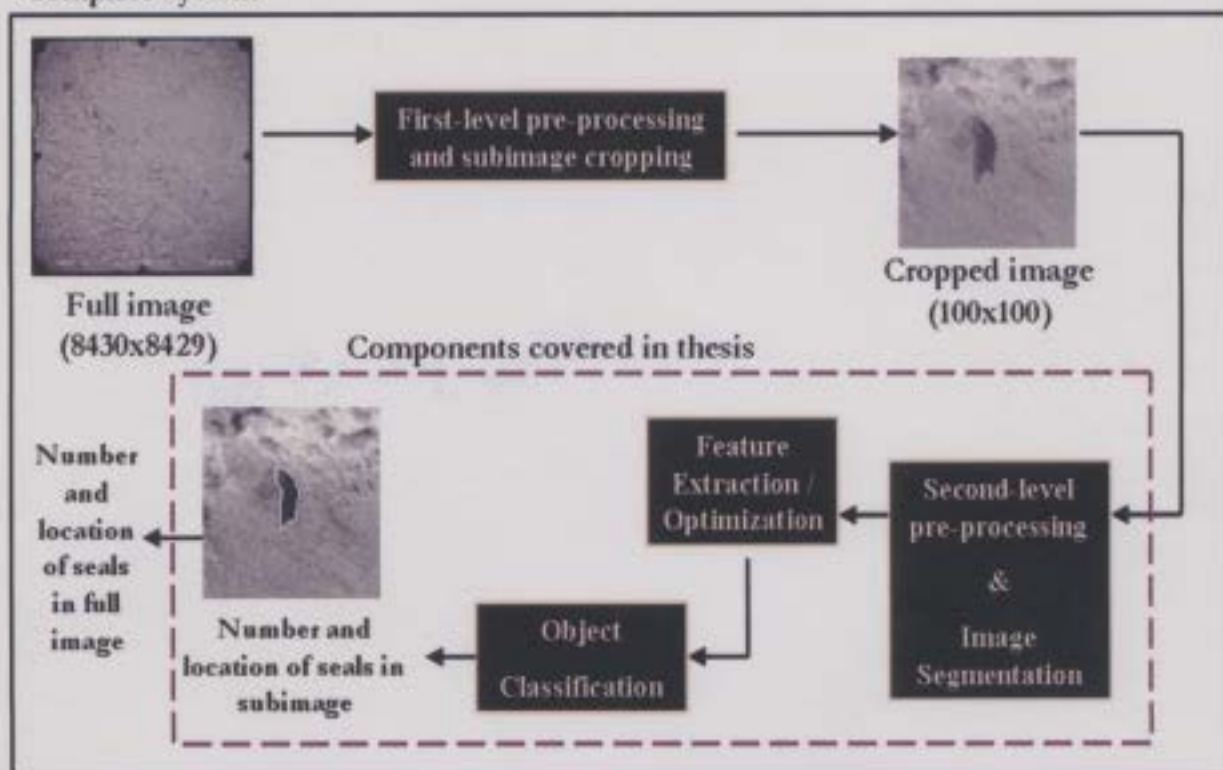


Figure 3.2: A complete software system for identifying and counting seal pups in aerial photographs. The components implemented in this research are contained within the dotted line.

For the purpose of this work, subimages were *manually* selected to ensure they contain specific seal pup data required to train and test the segmentation and clas-

sification algorithms. Subimages are  $100 \times 100$  pixels in size and were cropped such that each contains at least one seal pup (as identified in ground-truth images<sup>1</sup>) in an arbitrary position within the subimage. The majority of subimages were cropped by Hogan *et al.* [64] and graciously provided for this research. To produce a more complete set of test and training data, additional subimages were cropped using a MATLAB GUI designed by this author (Figure 3.3). A total of 900 subimages were produced. Since the data set is large, a simple random holdout validation strategy was used to separate training and test data. Three hundred subimages were chosen randomly to form the test data, and the remaining 600 subimages were retained to develop the segmentation algorithm and train the classifier.



Figure 3.3: MATLAB GUI used to crop  $100 \times 100$  pixel subimages from the large original images. The cropped area is highlighted in green on the main axis and displayed in the top right-hand axis.

<sup>1</sup>It is assumed that all seal pups manually identified by trained readers in ground-truth images provided for this research are correctly identified.

### 3.2.3 Challenges

Aerial images of whelping concentrations may contain a variety of features including open waterways, sea ice, shadows, adult seals, seal pups and other marine life. Sea ice is also varied in shape and texture; it may appear flat and smooth or rough and jagged. All of these features make it difficult to distinguish seal pups from their surroundings and must be carefully considered when developing the segmentation algorithm. As discussed in the previous section, regions known to be void of seal pups can be initially masked out to help reduce the complexity of the data and improve the segmentation results.

The segmentation algorithm must also be robust to complex conditions such as uneven illumination, shadows, occlusions, and objects grouped together. In some instances, seal pups and ice chunks are similarly shaped. In other cases, seal pups and the adjacent background pixels have similar gray levels. Due to a variety of objects and lighting conditions, it is highly unlikely that an automated segmentation algorithm will work perfectly for every image. The goal is to maximize the number of correctly segmented seal pups while eliminating as much background as possible.

The images presented in Table 3.1 represent a variety of conditions that must be addressed by the segmentation algorithm. As these examples show, the pixel values of seal pups vary significantly. In most cases, the background is composed of sea ice which has typically lighter gray levels than seal pups. This fact is used when developing the adaptive thresholding algorithm. However, ice chunks cast dark shadows which may occlude seal pups or appear to be connected to a seal object. In the latter case, the Isolate Connected Components algorithm attempts to separate adjacent objects that are inadvertently segmented as one object because they share similar intensity values.



Table 3.1: The segmentation algorithm must be robust to a variety of objects and lighting conditions.









Subimage	Condition
	A dark seal pup on a light background is trivial to segment.
	Seal pups and sea ice may cast dark shadows on the ice. A seal's shadow may be difficult to distinguish from its body.
	A light seal pup is difficult to distinguish from a light background (i.e. similar gray levels).
	A seal pup with homogeneous gray levels on a mixed background (i.e. adjacent background pixels are both light and dark in intensity).
Continued on next page	

Table 3.1 – continued from previous page

Subimage	Condition
	A light seal pup on a light background adjacent to dark non-seal pup objects. The seal pup may be eliminated as background.
	A dark seal pup adjacent to dark non-seal pup objects. These may be segmented as a single object.
	A seal pup partially occluded by other objects.
	A seal pup on a very complex background.

### 3.3 Segmentation Algorithm

The segmentation algorithm is broken down into 5 main steps:

1. Enhance the image to increase the dynamic range of gray levels.
2. Apply adaptive thresholding using between-class variance and histogram skewness.
3. Apply a “cleaning” algorithm to the threshold image.
4. Separate connected objects using the Isolate Connected Components (ICC) algorithm.
5. Remove outlier objects using select features.

These steps are fully explained in the following sections.

#### 3.3.1 Image Enhancement

Image enhancement techniques are used to improve an image, where “improve” is sometimes defined objectively (e.g. increase the signal-to-noise ratio), and sometimes subjectively (e.g. make certain features easier to see by modifying the colors or intensities) [5]. Contrast stretching is an image enhancement technique that attempts to improve the contrast in an image by “stretching” the range of intensity values it contains to span a desired range of values. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even incorrectly setting a lens aperture during image acquisition [68].

Figure 3.4 shows a typical piecewise linear transformation function  $T(r)$  used for contrast stretching. The locations of points  $(r_1, q_1)$  and  $(r_2, q_2)$  control the shape of the transformation. In general,  $r_1 \leq r_2$  and  $q_1 \leq q_2$  is assumed so that the function



is single valued and monotonically increasing. This condition preserves the order of gray levels, thus preventing the creation of intensity artifacts in the processed image [68]. To map intensity values from  $[r_1, r_2]$  to  $[q_1, q_2]$ , the following equation should be applied to each input image pixel with values in the range  $[r_1, r_2]$  :

$$I_{out}(x, y) = (I_{in}(x, y) - r_1) \left( \frac{q_2 - q_1}{r_2 - r_1} \right) + q_1, \quad (3.1)$$

where  $I_{in}$  is the input image,  $I_{out}$  is the output image, and  $I(x, y)$  represents the gray level of an image pixel at image coordinates  $(x, y)$ . Pixel values in  $I_{in}$  below  $r_1$  and above  $r_2$  are typically clipped; that is, values below  $r_1$  are mapped to  $q_1$ , and those above  $r_2$  are mapped to  $q_2$ .

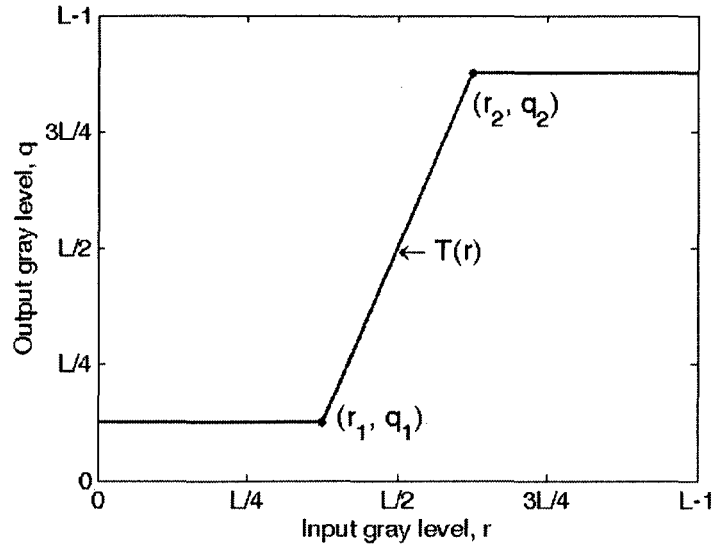
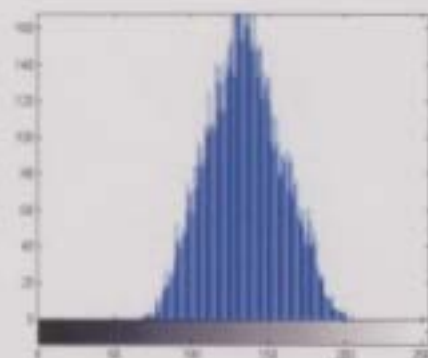


Figure 3.4: Typical transformation used for contrast stretching.  $L$  is the number of intensity levels.

To illustrate contrast stretching, consider the subimage in Figure 3.5a and the corresponding histogram in Figure 3.5b. Notice how the intensity range is rather narrow. It does not cover the potential gray-scale range of  $[0, 255]$  and is missing the high and low values that would result in good contrast. If we let  $(r_1, q_1) = (r_{\min}, 0)$



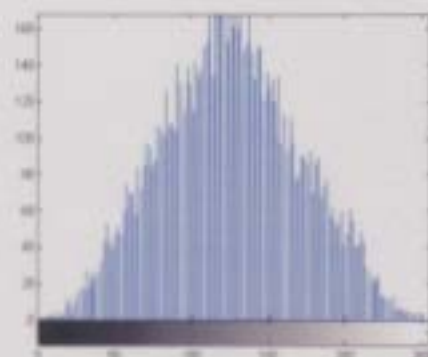
(a)



(b)



(c)



(d)

Figure 3.5: (a) Example of low-contrast image. (b) Histogram of (a) shows narrow intensity range. (c) Image after contrast stretching. (d) Histogram of (c) shows increased dynamic range.

and  $(r_2, q_2) = (r_{\max}, 255)$ , where  $r_{\min}$  and  $r_{\max}$  denote the minimum and maximum gray levels in the input image, applying Equation (3.1) stretches the levels from their original range to the full gray-scale range  $[0, 255]$ . Figures 3.5c and 3.5d show the results. Increasing the dynamic range of pixel values to the entire gray-scale range highlights contrast between lighter and darker regions. This helps to make seal pups easier to “see” by the segmentation algorithm. Stretching intensity values to the full gray-scale range is applied to all subimages as a pre-processing step.

### 3.3.2 Adaptive Thresholding and Between-Class Variance

The most trivial thresholding technique is to partition an image histogram by using a single global threshold,  $T$ . Segmentation is then accomplished by scanning the image pixel-by-pixel and labeling each pixel as object or background, depending on whether the gray level of that pixel is greater or less than the value of  $T$ . Imaging factors such as uneven illumination often prevent an image from being partitioned effectively by a single global threshold. An alternative approach is to divide the original image into subimages and then utilize a different threshold to segment each subimage. The key issues in this approach are how to subdivide the image and how to estimate the threshold for each resulting subimage. Since the threshold used for each pixel depends on the location of the pixel in terms of the subimages, this approach is called *adaptive thresholding* [68].

To perform adaptive thresholding on the seal pup data, each pre-processed  $100 \times 100$  pixel subimage is subdivided into nine regions by placing a  $3 \times 3$  grid with 50% overlap over the image. Each of the nine resulting overlapped regions is  $50 \times 50$  pixels in size. An illustration of this subdivision is shown in Figure 3.6.

The next step is to compute a threshold value  $T$  for each subdivided region. The main objective is to select the value of  $T$  that minimizes the average error in making the decision that a given pixel belongs to an object or to the background. This value is then called the *optimal threshold*.

Between-class variance, first introduced by Otsu [72], is a discriminant function used to determine an optimal threshold from an image histogram in order to segment the image into nearly uniform regions. This function has been used in previous research [73, 74] and reported to perform best in a survey of thresholding techniques [75]. As shown by [73], the between-class variance can also be used as a measure of

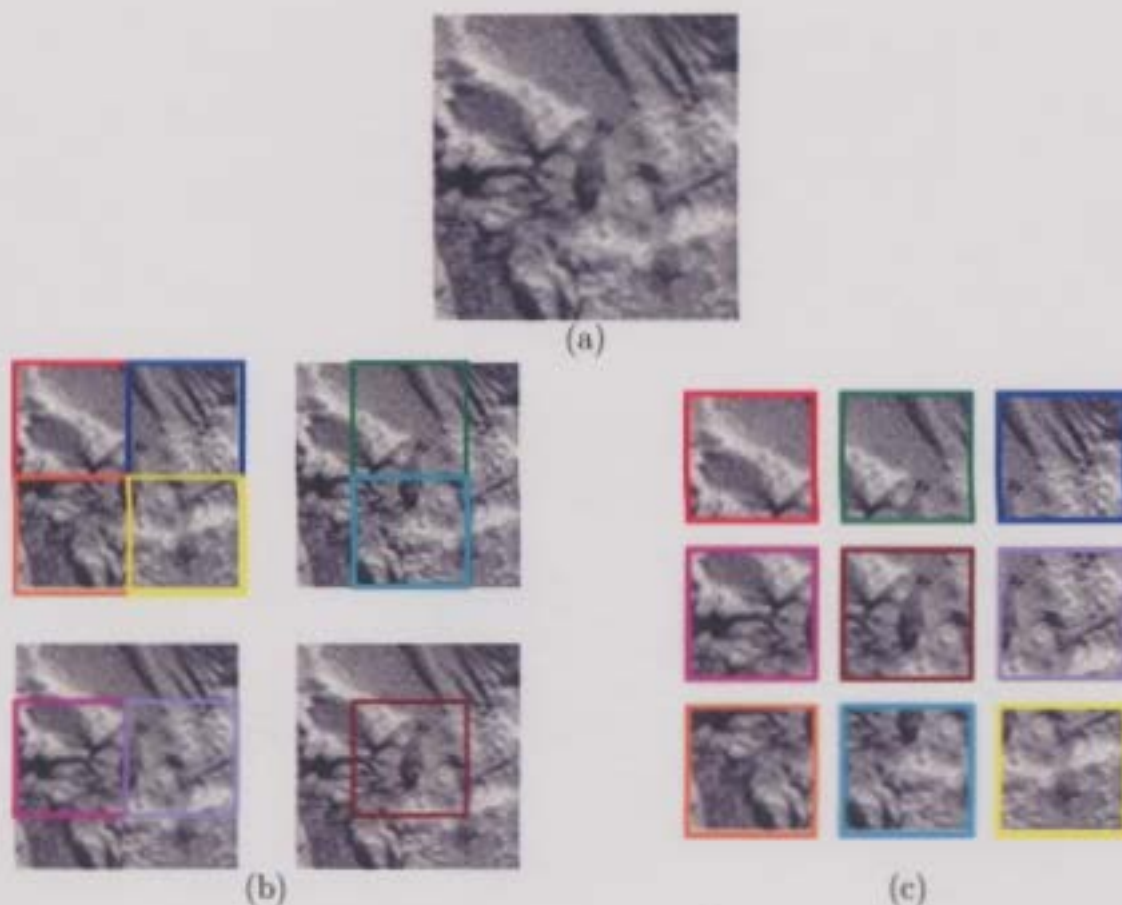


Figure 3.6: (a) Enhanced  $100 \times 100$  pixel subimage. (b) Image in (a) is subdivided into nine regions (illustrated with colored squares) by placing a  $3 \times 3$  grid with 50% overlap over the image. (c) Each of the nine resulting overlapped regions is  $50 \times 50$  pixels in size.

image bimodality. For this application, if the image histogram of a  $50 \times 50$  subdivided region is determined to be bimodal, between-class variance is used to determine the optimal threshold for this region. Otherwise, the skewness of the image histogram about the mean is used to determine the optimal threshold.

### Computation of Between-Class Variance and Image Bimodality

For each  $50 \times 50$  pixel subdivided region, the gray-level histogram is normalized and regarded as a discrete probability function  $p(i)$  such that

$$p(i) = \frac{n_i}{M}, \quad p(i) \geq 0 \quad \text{and} \quad \sum_{i=0}^{255} p(i) = 1 \quad (3.2)$$

where  $n_i$  is the frequency of the gray level  $i$  and  $M$  is the total number of pixels in the image. Each pixel in the image assumes a gray-level value from the set  $[0, 1, 2, \dots, 255]$ . If the histogram is divided into two classes by the gray-level intensity  $t$ , then the probabilities of the respective classes can be expressed as

$$p_1(t) = \sum_{i=0}^t p(i) \quad , \quad p_2(t) = \sum_{i=t+1}^{255} p(i) \quad (3.3)$$

Similarly, the class means  $m_1$  and  $m_2$  are given by

$$m_1(t) = \sum_{i=0}^t i \cdot \frac{p(i)}{p_1(t)} \quad , \quad m_2(t) = \sum_{i=t+1}^{255} i \cdot \frac{p(i)}{p_2(t)} \quad (3.4)$$

The class variances are then given by

$$\sigma_1^2(t) = \sum_{i=0}^t (i - m_1)^2 \cdot \frac{p(i)}{p_1(t)} \quad , \quad \sigma_2^2(t) = \sum_{i=t+1}^{255} (i - m_2)^2 \cdot \frac{p(i)}{p_2(t)} \quad (3.5)$$

The total variance is defined as

$$\sigma_T^2 = \sigma_W^2 + \sigma_B^2 \quad (3.6)$$

where  $\sigma_W^2$  (*within-class variance*) and  $\sigma_B^2$  (*between-class variance*) are expressed as

$$\sigma_W^2(t) = p_1(t) \cdot \sigma_1^2(t) + p_2(t) \cdot \sigma_2^2(t), \quad (3.7)$$

$$\sigma_B^2(t) = p_1(t) \cdot (m_1(t) - m_T)^2 + p_2(t) \cdot (m_2(t) - m_T)^2 \quad (3.8)$$

where  $m_T$  is the mean pixel value of the entire image.

Within-class variance is the sum of the individual class variances weighted by their respective class probabilities. Between-class variance is an indicator of the “distance” between the class modes. It provides valuable information as to how close the two classes are to each other. It can also be expressed in terms of class probabilities and means only:

$$\sigma_B^2(t) = p_1(t) \cdot p_2(t) [m_1(t) - m_2(t)]^2 \quad (3.9)$$

The optimal threshold  $t^*$ , which segments the image into nearly uniform regions, is the gray level at which between-class variance is maximum:

$$\sigma_B^2(t^*) = \max_{0 \leq t < 255} \sigma_B^2(t). \quad (3.10)$$

The same optimum threshold can be obtained by minimizing within-class variance since their sum is constant (refer to Equation (3.6)). However, Equation (3.9) is computationally less expensive than Equation (3.7) as it does not include class variances. In the sequential search for the optimal threshold, the class probabilities and means can be progressively computed to reduce computation time [73].

In image segmentation, the detection of image bimodality becomes necessary to make intelligent decisions prior to segmentation. In this application, image bimodality is used to determine the method by which to compute the optimal threshold. The term bimodal indicates a statistical distribution having two separated peaks or two local maxima. The optimal threshold is the one that best separates these two distinct regions of the histogram (Figure 3.7a). An image histogram may also be unimodal

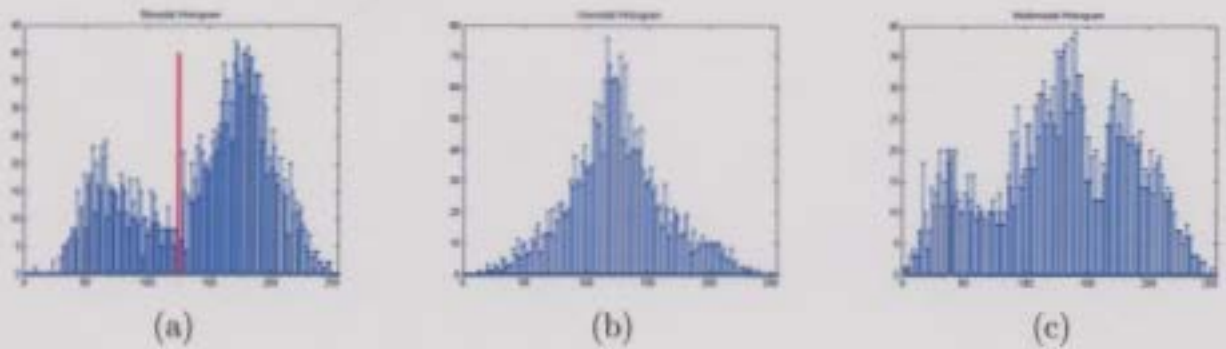


Figure 3.7: (a) Bimodal distribution with optimal threshold indicated by red line. (b) Unimodal distribution. (c) Multimodal distribution with more than two peaks. An optimal threshold may be more difficult to determine for distributions shown in (b) and (c).

(have a single peak or local maximum) or multimodal with more than two peaks. In these cases, an optimal threshold may be more difficult to determine (Figures 3.7b and 3.7c).

The normalized between-class variance,

$$b(t) = \frac{\sigma_B^2(t)}{\sigma_{total}^2} = \frac{p_1(t) \cdot p_2(t) [m_1(t) - m_2(t)]^2}{\sigma_{total}^2} \quad (3.11)$$

is used to detect bimodality. That is, the maximum  $b(t)$

$$b_{max} = \max_{0 \leq t < 255} b(t) \quad (3.12)$$

is called the *bimodality coefficient* and is used as an indicator of bimodality. The  $b(t)$  takes on values between zero and one. One difficulty in the bimodality test is to select a threshold value for  $b_{max}$ . For balanced uniform distributions, the theoretical value of  $b(t)$  is 0.75 [73]. That is,  $b_{max} \geq 0.75$  indicates bimodality. However, different approaches can be taken depending on the application. In actuality, region distributions overlap significantly. For images consisting of regions with highly overlapping distributions, a value less than the theoretical value can be selected as the threshold.

For this application,  $b_{\max} \geq 0.71$  was empirically chosen to indicate bimodality.

To illustrate, consider the  $100 \times 100$  pixel subimage of a seal pup in Figure 3.8; two  $50 \times 50$  subdivided regions are highlighted. The histogram and corresponding bimodality coefficient of each subdivided region are also shown. The histogram of the subdivided region in Figure 3.8b has a bimodal distribution with  $b_{\max} = 0.78$ . The histogram of the subdivided region in Figure 3.8d is clearly not bimodal with  $b_{\max} = 0.66$ .

If a subdivided region is found to have a bimodal distribution (i.e.  $b_{\max} \geq 0.71$ ), then the optimal threshold is  $t^*$  (i.e. the value of  $t$  that maximizes the between-class variance). As discussed previously, this threshold value segments the image into nearly uniform regions. If the subdivided region does not have a bimodal distribution (i.e.  $b_{\max} < 0.71$ ), an optimal threshold value is computed based on the skewness of the image histogram.

### Skewness of Image Histogram

Skewness is a measure of the asymmetry of the data around the sample mean. It is written as

$$\gamma = \frac{\mu_3}{\sigma_3} \quad (3.13)$$

where  $\mu_3$  is the third moment about the mean and  $\sigma$  is the standard deviation. A negative skewness,  $\gamma < 0$ , indicates the data are spread out more to the left of the mean than to the right. If skewness is positive,  $\gamma > 0$ , the data are spread out more to the right. The skewness of the normal distribution, or any perfectly symmetric distribution, is zero ( $\gamma = 0$ ).

For computing the optimal threshold, it is important to note that background pixels are typically lighter in intensity than seal pup pixels. Therefore, the threshold



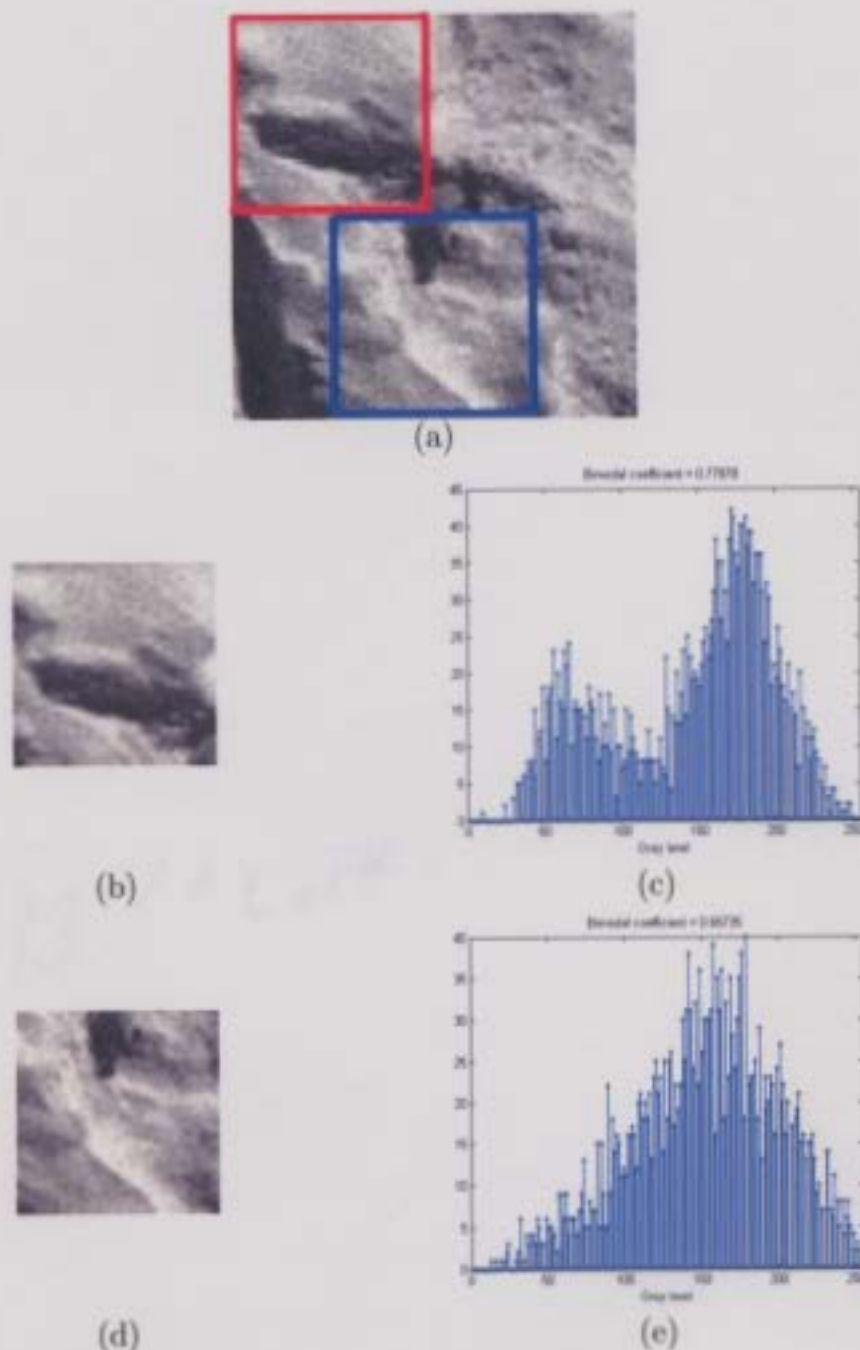


Figure 3.8: (a)  $100 \times 100$  pixel gray-scale image of seal pup. Two  $50 \times 50$  subdivided regions are highlighted to illustrate bimodality. (b)  $50 \times 50$  subdivided region highlighted in red in (a). (c) Histogram of (b);  $b_{\max} = 0.78$  indicates that image histogram is bimodal. (d)  $50 \times 50$  subdivided region highlighted in blue in (a). (e) Histogram of (d);  $b_{\max} = 0.66$  indicates that image histogram is not bimodal.

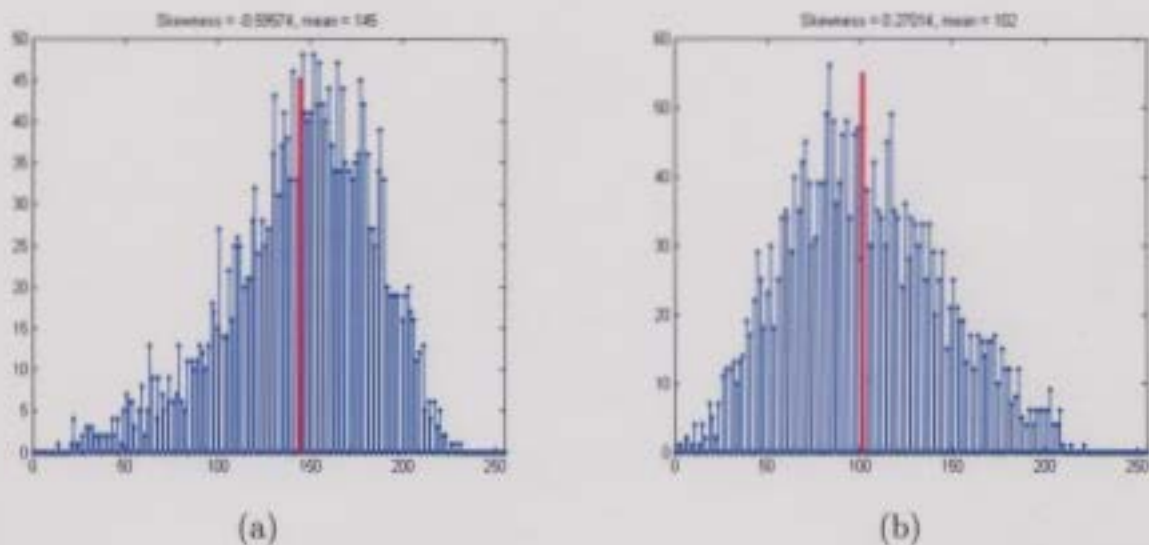


Figure 3.9: (a) Negative skewness; data are spread out more to the left of the mean. (b) Positive skewness; data are spread out more to the right of the mean. In both figures above, the mean is indicated by a red vertical line.

will be biased toward darker gray levels in order to maximize the number of light intensity pixels labeled as background. If darker background pixels are incorrectly labeled as foreground during thresholding, such as areas comprised of shadows, these pixels may still be labeled as background, or belonging to outlier objects, in a later stage of the algorithm.

For image histograms with a negative skewness (Figure 3.9a), the left tail of the distribution is the longest and the mass of pixel values is concentrated on the right of the histogram (i.e. toward lighter pixel values). In this case, the histogram peak is usually located to the right of the mean pixel value. A large mass of light intensity values typically indicates a large area of lighter background pixels, most likely sea ice, with minimal darker objects. Seal pup pixels are more likely to be included among the darker pixel values. In order to maximize the number of lighter pixels labeled as background, the threshold value is chosen as the mean pixel value,  $\mu$ , minus one standard deviation,  $\sigma$ , of all pixel values in the subdivided region:

$$T_{\gamma < 0} = \mu - \sigma \quad (3.14)$$

For image histograms with a positive skewness (Figure 3.9b), the right tail of the distribution is the longest and the mass of pixel values is concentrated on the left of the histogram (i.e. toward dark pixel values). In this case, the histogram peak is usually located to the left of the mean pixel value. A larger mass of dark intensity values typically indicates the presence of more shadows with a smaller area of lighter background pixels. Seal pups may be among the darker pixels values, but they are not likely to be as dark as shadows which tend to be black. Therefore, the threshold value is more cautiously chosen as the mean (minus one standard deviation is not applied) to decrease the likelihood of eliminating the seal as background. For normally distributed histograms the skewness is zero. In this case, the optimal threshold is also chosen as the mean pixel value:

$$T_{\gamma \geq 0} = \mu \quad (3.15)$$

### **Final Threshold Image**

Summarizing the previous discussion, the optimal threshold  $T$  for each  $50 \times 50$  pixel subdivided region is given by

$$T = \begin{cases} t^* & \text{if } b_{\max} \geq 0.71 \\ \mu - \sigma & \text{if } b_{\max} < 0.71 \text{ and } \gamma < 0 \\ \mu & \text{if } b_{\max} < 0.71 \text{ and } \gamma \geq 0 \end{cases} \quad (3.16)$$

After the threshold is determined for a subdivided region, a binary image is created such that all background pixels are 0-valued and all foreground pixels are 1-valued.

In other words, given an input image  $I$ , the threshold image  $I_T$  is defined by

$$I_T(x, y) = \begin{cases} 1 & \text{if } I(x, y) < T \quad (\text{foreground}) \\ 0 & \text{if } I(x, y) \geq T \quad (\text{background}) \end{cases} \quad (3.17)$$

where  $I(x, y)$  represents the gray level of an image pixel at image coordinates  $(x, y)$ . The resulting threshold images for the nine subdivided regions in Figure 3.6c are shown in Figure 3.11a.

As a result of the 50% overlap between subdivided regions, certain pixels in the  $100 \times 100$  subimage will belong to more than one of these regions (see Figure 3.10). A



Figure 3.10: Pixels in the shaded red area will belong to more than one subdivided region during thresholding.

pixel in this overlap area may be labeled as foreground in one subdivided region and background in another. In order to determine a single label for this pixel in the final threshold image, a bitwise OR operation [68] is applied to all labels assigned to this pixel. Bitwise OR sets a bit (or in this case, a binary pixel value) to 1 if one or both of the corresponding bits in its operands are 1, and to 0 if both of the corresponding bits are 0. In terms of pixel values, if a pixel is labeled as foreground in any of the subdivided regions, the bitwise OR operation ensures it will be labeled as foreground (i.e. possible target object) when the image is reassembled. A pixel will be labeled

as background in the final threshold image if and only if it is labeled as background in all subdivided regions. The final threshold result for the image in Figure 3.6a is shown in Figure 3.11b.

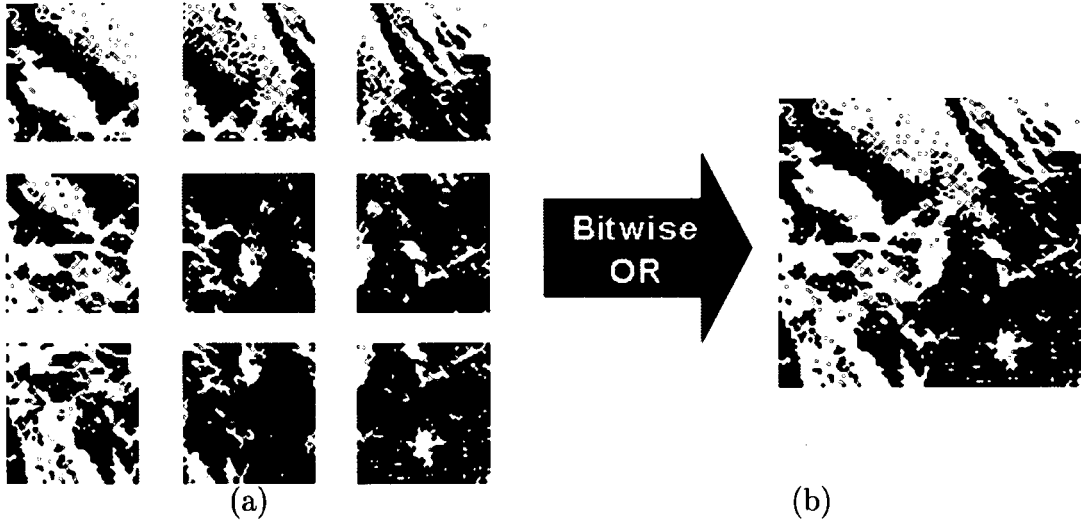


Figure 3.11: (a) The resulting binary images for the nine subdivided regions in Figure 3.6c. (b) A bitwise OR operation is performed on the nine subdivided regions to produce the final threshold image.

### 3.3.3 “Cleaning” Algorithm

The binary image produced from the adaptive thresholding algorithm contains extraneous pixels and appears a little “messy” (Figure 3.11b). It may be difficult to distinguish possible target objects from this image. Therefore, a “cleaning” algorithm is applied to the threshold image using edge detection, line dissection, and removal of objects based on size constraints.

First, Canny edge detection [76] is applied to the enhanced image produced in step one of the segmentation algorithm. The Canny method finds edges by looking for local maxima of the image gradient which is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges.

Weak edges are included in the output only if they are connected to strong edges. This method is therefore less likely to be fooled by noise than other edge detection techniques and more likely to detect true weak edges. Figure 3.12 shows the enhanced  $100 \times 100$  pixel gray-scale image and its corresponding binary edge image after the Canny edge detection is applied.

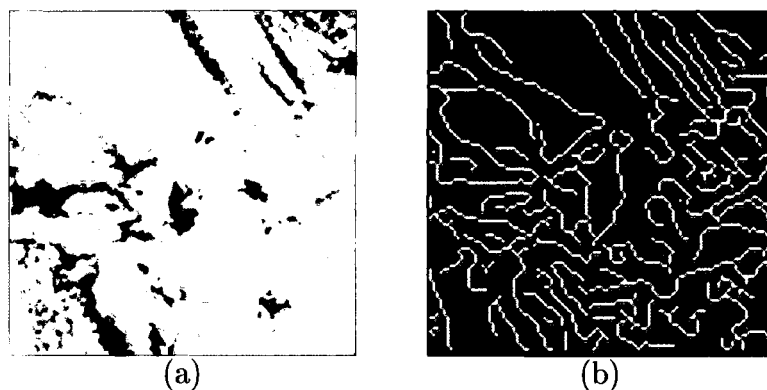


Figure 3.12: (a) Enhanced gray-scale image. (b) Edge image produced from Canny edge detection.

Next, the negative of the Canny edge image is obtained by reversing 0-valued (black) and 1-valued (white) pixels (see Figure 3.13a). A bitwise AND operation [68] is applied between the negative Canny image and the binary threshold image produced in step two of the segmentation algorithm (see Figure 3.13b). Bitwise AND sets a bit (or a binary pixel value) to 0 if one or both of the corresponding bits in its operands are 0, and to 1 if both of the corresponding bits are 1. The AND operation superimposes the negative Canny edge image on the threshold image producing clearly outlined objects (see Figure 3.13c). This process also helps to disconnect adjacent objects.

To further clean the image and separate adjacent objects, an algorithm was designed to remove isolated pixels (individual 1's that are surrounded by 0's) and dissect horizontal and vertical lines that are one pixel in width. Table 3.2 gives examples of

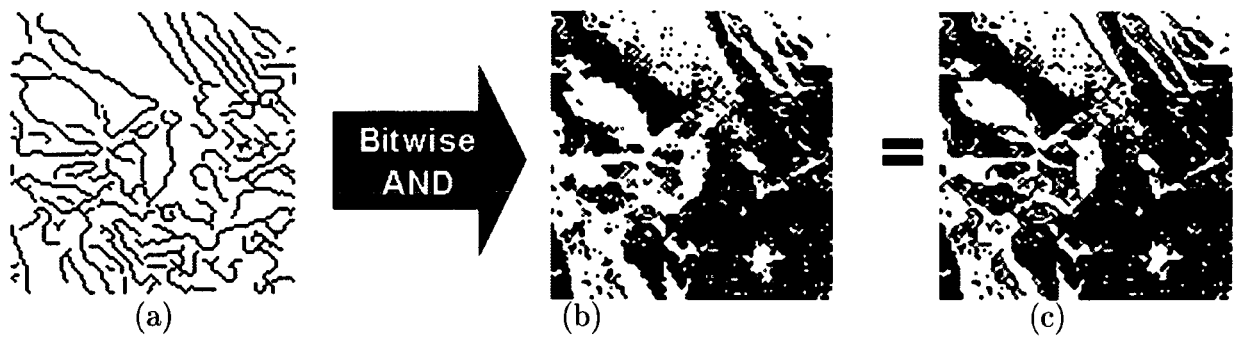


Figure 3.13: (a) Negative of binary Canny edge image. (b) Bitwise AND operation is performed between the negative Canny edge image and the threshold image. (c) The result is a threshold image with clearly outlined objects.

applying these operations to specific pixel patterns. Figure 3.14a shows the result of applying the operations to the image in Figure 3.13c.

Table 3.2: As part of the “cleaning” process, isolated pixels are removed, and horizontal and vertical lines are dissected. This table shows examples of applying these operations to specific pixel patterns.

Pixel Pattern	Operation	Resulting Pixel Pattern
0 0 0 0 1 0 0 0 0	remove isolated pixels	0 0 0 0 0 0 0 0 0
0 1 1 0 1 0 1 1 0	break vertical lines 1-pixel in width	0 1 1 0 0 0 1 1 0
0 0 1 1 1 1 1 0 0	break horizontal lines 1-pixel in width	0 0 1 1 0 1 1 0 0

The final step in this cleaning process is to compute the area of each object in the threshold image shown in Figure 3.14a. Area is defined as the actual number of pixels that compose an object. In order to distinguish individual objects, the type of object connectivity must be defined. There are two standard two-dimensional connectivity types: 4-connected and 8-connected. Four-connected pixels are connected if their edges touch. This means that a pair of adjoining pixels are part of the same

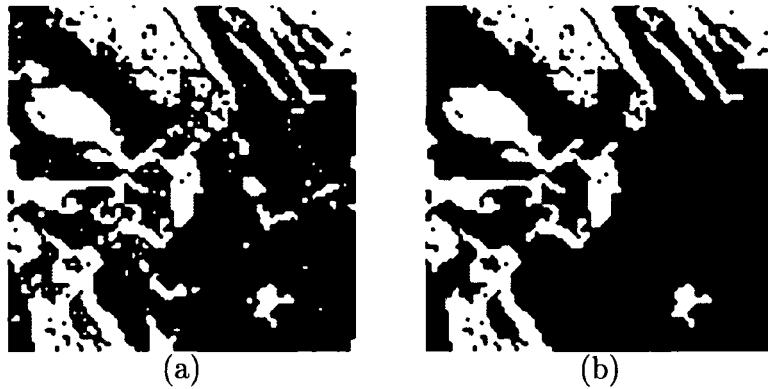


Figure 3.14: (a) A “cleaning” process is applied that removes isolated pixels and dissects horizontal and vertical lines that are 1-pixel in width. (b) A clean binary image is produced when objects with an area less than 50 are removed from the threshold image in (a).

object if they are both “on” (i.e. 1-valued) and are connected along the horizontal or vertical direction. Eight-connected pixels are connected if their edges or corners touch. This means that if two adjoining pixels are “on”, they are part of the same object, regardless of whether they are connected along the horizontal, vertical, or diagonal direction. The type of connectivity chosen affects the number of objects found in an image and the boundaries of those objects. For computing area, objects are defined as 4-connected. Objects with an area less than 50 are removed since these are too small to be considered seal pups. This quantity was empirically chosen through the training process. The result is a clean binary image (Figure 3.14b) to which the Isolate Connected Components algorithm is applied.

### 3.3.4 Isolate Connected Components

In Figure 3.14b, some regions that appear as a large single object are, in fact, multiple smaller objects connected together. This may occur if adjacent objects have similar gray levels. For example, in Figure 3.15, a seal pup and part of the adjacent ice are segmented as one object because they share similar intensity values. In some



instances, two or more seal pups lying adjacent to each other on the ice may be segmented as a single object. Neighboring objects may also remain connected if edges produced from the Canny algorithm are incomplete (i.e. some edges are not detected). An example of this situation is illustrated in Figure 3.16. The Isolate Connected Components (ICC) algorithm attempts to separate grouped objects into their constituent parts, or components, by breaking narrow isthmus regions that connect them.



Figure 3.15: (a) Two objects (seal pup and adjacent ice) with similar gray levels are segmented as one object. (b) Original gray-scale image with outline of objects.



Figure 3.16: (a) Neighboring objects remain connected because the Canny algorithm did not detect all edges (i.e. incomplete edge detection). (b) Original gray-scale image.

The ICC algorithm should be applied to each 4-connected object in the binary

image produced in the previous step. Just one of the objects in Figure 3.14b is used here to illustrate the algorithm (see Figure 3.18a). The first step is to apply a morphological closing operation to remove all holes that are 1 pixel in size (see Figure 3.18b). Larger holes are not closed because they represent a more significant ‘vacant’ area inside an object that may aid in isolating or separating connected components.

The next step of the algorithm depends on the area of the object. Figure 3.17 shows the distribution of area values for seal pup objects segmented from training data. While area values widely range from 62 to 382, over 75% of seal pup objects

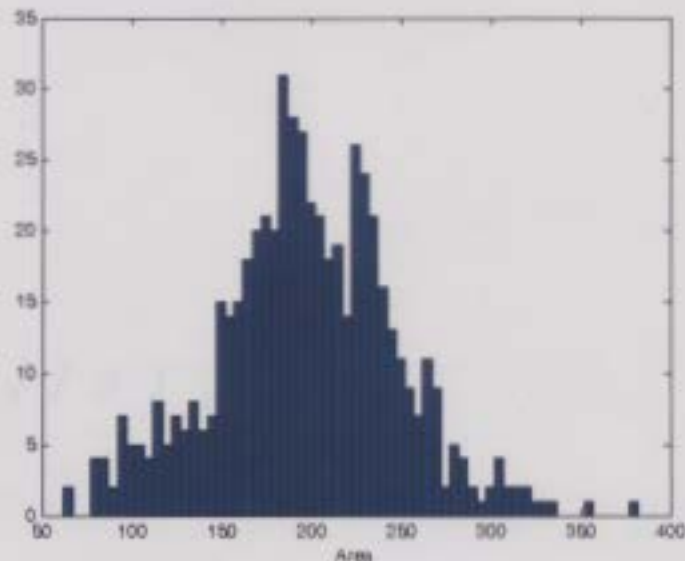


Figure 3.17: Distribution of area values for seal pup objects segmented from training data.

have an area less than 230. Therefore, this value was empirically chosen as a threshold for determining how to proceed with the ICC algorithm. Objects with an area greater than or equal to 230 are less likely to be seal pups. Therefore, the ICC algorithm will attempt to separate these objects into possible constituent parts. The algorithm will *not* attempt to divide objects with an area less than 230 because this may have the undesired effect of breaking a seal pup into multiple parts. For example, the body

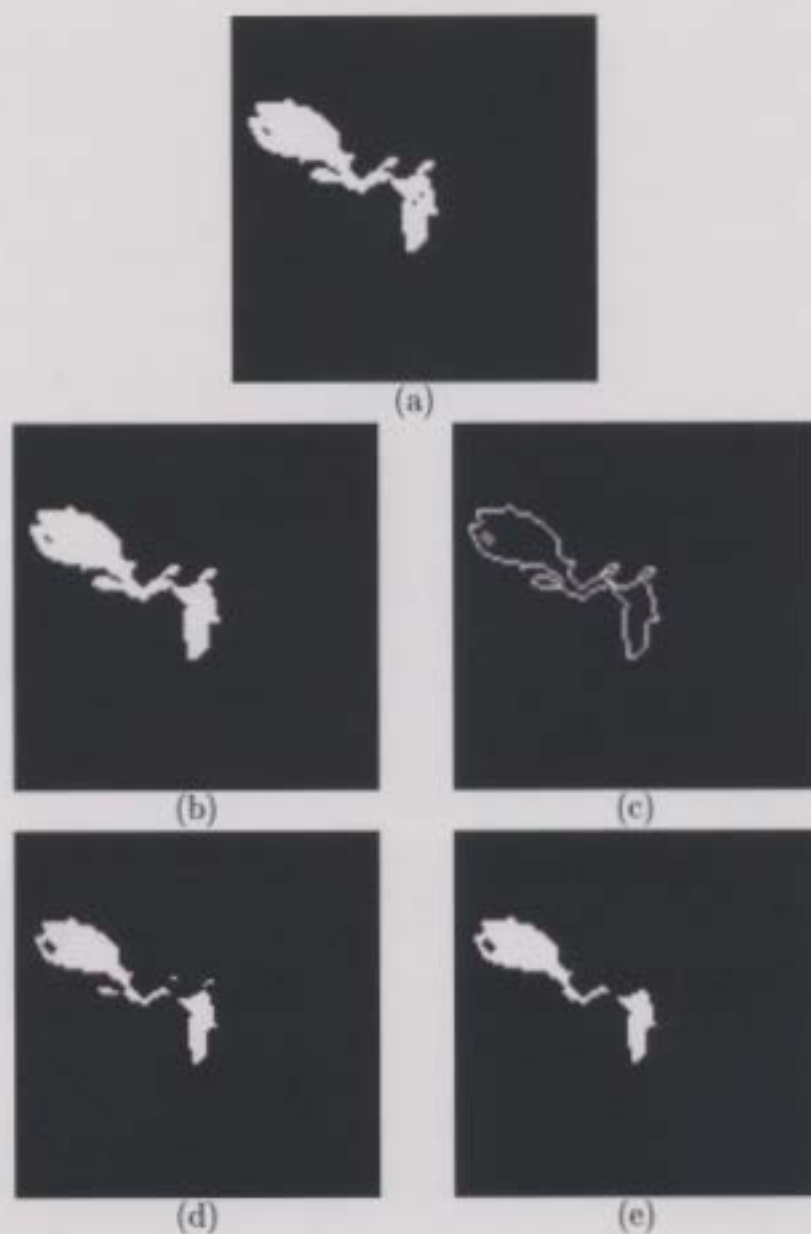


Figure 3.18: (a) Object from Figure 3.14b used to illustrate ICC algorithm. (b) Morphological closing is applied to remove holes 1 pixel in size and area of object is computed (area = 599). (c) Binary image of object perimeter. (d) Perimeter is subtracted from object (i.e. (b) minus (c)). This divides the object into parts by removing narrow isthmus regions that are at most 2 pixels in width. (e) Extraneous blobs with area less than 15 are removed resulting in two isolated component objects.

and tail flippers of a seal pup object may be separated if they are connected by a narrow isthmus. Therefore, objects that have an area less than 230 remain whole. The only process applied to these objects is a morphological closing operation using a disk-shaped structuring element with radius 2. This attempts to close any remaining interior holes.

For each object with area greater than or equal to 230, the next step is to remove the object perimeter. A pixel is part of the perimeter if it is nonzero and it is connected to at least one zero-valued pixel. The removal process is executed by creating a binary image of the perimeter (Figure 3.18c) and subtracting it from a binary image of the object (Figure 3.18b). This effectively divides the object into smaller parts, referred to hereafter as component objects, by removing narrow isthmus regions that are at most 2 pixels in width (Figure 3.18d). Any resulting extraneous blobs with area less than 15 are removed (Figure 3.18e).

If the area of a resulting component object is still greater than or equal to 230, the perimeter removal process is repeated on that object. For example, the area of the left and right component objects in Figure 3.18e is 286 and 111, respectively. The left component object is processed first; since its area is greater than 230, the ICC algorithm will attempt to divide it into even smaller components by subtracting its perimeter a second time. As illustrated in Figure 3.19, the process is similar to the previous step. First, a binary image of the object perimeter is created and subtracted from a binary image of the object. This attempts to divide the object into smaller components by removing narrow isthmus regions that are at most 2 pixels in width. Any resulting extraneous blobs with area less than 10 are removed. For this particular example, the division did not result in multiple component objects. However, if multiple objects were produced, the perimeter removal process is not applied a third time, regardless of object size. We are only interested in separating

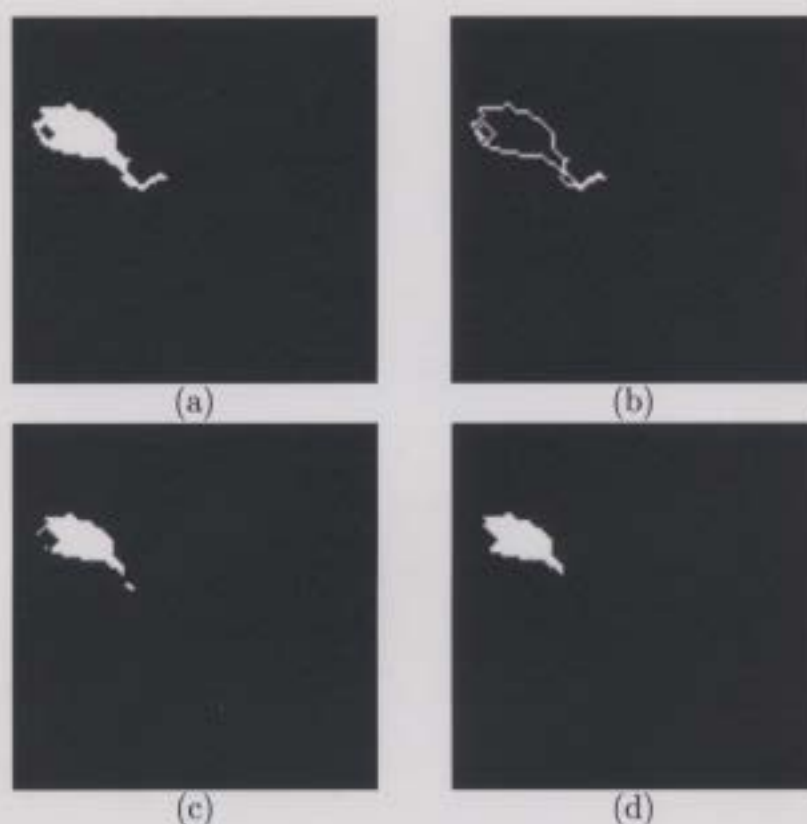


Figure 3.19: (a) Binary image of component object on left-hand side of Figure 3.18e. The area of this object is 286. (b) Binary image of object perimeter. (c) Perimeter is subtracted from object (i.e. (a) minus (b)). This divides the object into parts by removing narrow isthmus regions that are at most 2 pixels in width. (d) Extraneous blobs with area less than 10 are removed resulting in one component object.

components that are connected by *narrow* isthmus regions; applying the perimeter removal process twice will break isthmus regions that are at most 4 pixels in width. Applying the process a third time would break wider isthmus regions (6 pixels in width) and may even remove small, but significant, peninsula regions; this may have the undesired effect of breaking a single object into multiple parts or distorting the true shape of the object.

Next, the two perimeter layers that were removed must be reattached to the object in Figure 3.19d without reconnecting component objects (i.e. components must remain distinct). The perimeter layers must be reattached in the reverse order



they were removed. That is, the perimeter in Figure 3.19b is added first followed by the perimeter in Figure 3.18c.

To facilitate reattaching the perimeter, integer values are used to differentiate perimeter and object pixels. A value of 1 is assigned to pixels that compose the perimeter and unique integer values  $\geq 2$  are assigned to pixels that compose each object. Each integer label also corresponds to a different color in RGB space, so object and perimeter pixels can be easily viewed. Any perimeter pixel that is 8-connected to an object pixel is assigned the same integer label as the object (i.e. becomes an object pixel). A close-up look at this procedure is illustrated in Figure 3.20. It is important to note that if two or more objects share perimeter pixels, then the “first come, first serve” rule is applied. That is, perimeter pixels are assigned to objects in the order they are processed.

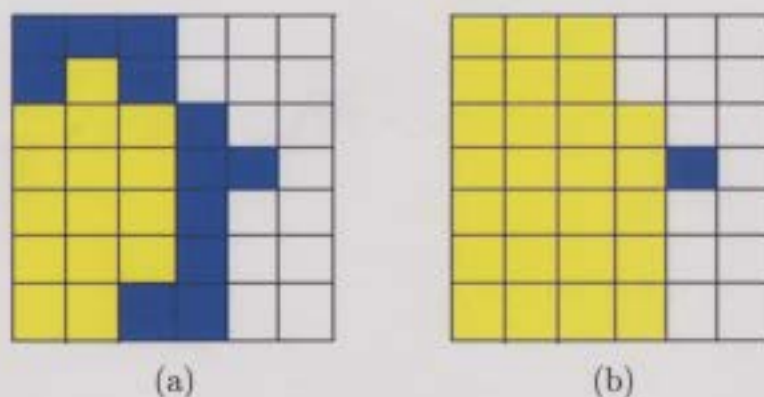


Figure 3.20: Close-up of pixels. (a) Object pixels are shown in yellow. Perimeter pixels are shown in blue. (b) Any perimeter pixel that is 8-connected to an object pixel is assigned the same integer label as the object (i.e. becomes an object pixel). The remaining blue pixel is the only perimeter pixel not 8-connected to an object pixel.

First, the perimeter in Figure 3.19b and the object in 3.19d are considered. Perimeter pixels are assigned a value of 1 and object pixels are assigned a value of 2. Figure 3.21a illustrates this pixel labeling in RGB space. The result of reattaching the

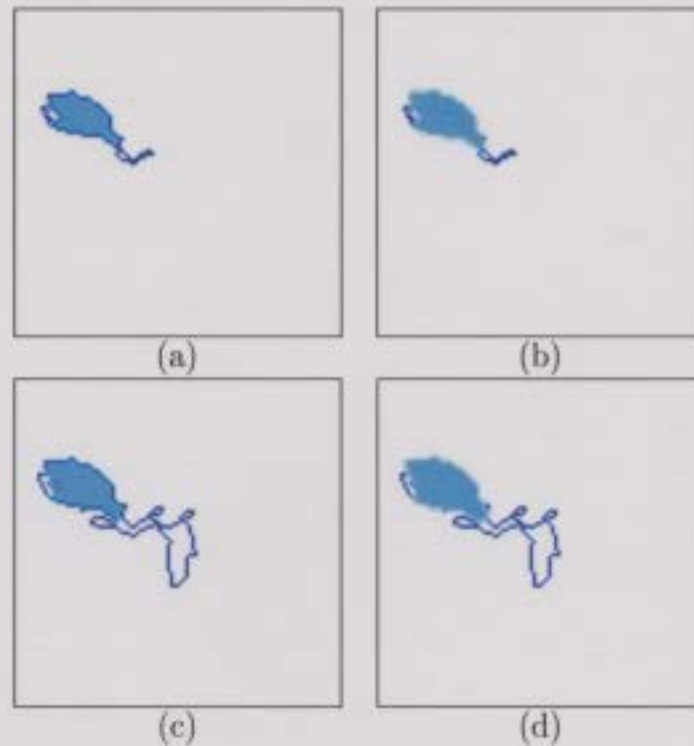


Figure 3.21: (a) Integer labels allow component objects and their perimeter to be viewed in RGB space. The perimeter in Figure 3.19b is shown in blue. The component object in Figure 3.19d is shown in cyan. (b) The perimeter pixels are reattached to the object using the 8-connected rule. (c) The perimeter in Figure 3.18c is shown in blue. The object in (b) is shown in cyan. (d) The perimeter pixels are reattached to the object using the 8-connected rule.

perimeter pixels using the 8-connected rule is shown in Figure 3.21b. Perimeter pixels not assigned to the object are labeled as background. Next, the perimeter in Figure 3.18c is reattached to the object using the 8-connected rule. Again, the perimeter pixels are assigned a value of 1 while the object pixels already have a value of 2. Figure 3.21c illustrates this pixel labeling in RGB space. The result of reattaching the perimeter pixels is shown in Figure 3.21d.

Next, the right component object in Figure 3.18e is processed. Recall that its area is 111. Therefore, the perimeter removal process is *not* repeated on this object. However, the perimeter in Figure 3.18c must still be reattached. The perimeter pixels have already been assigned a value of 1 from the previous step. The object pixels are

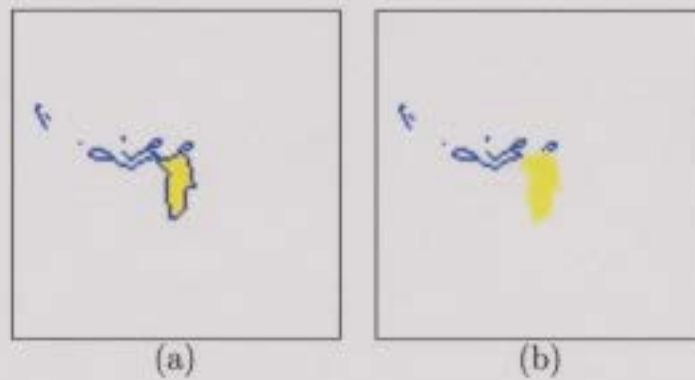


Figure 3.22: (a) The right-hand component object in Figure 3.18e is shown in yellow. The perimeter in Figure 3.18c is shown in blue. Note that some perimeter pixels are missing; these have already been assigned to another object. (b) The perimeter pixels are reattached to the object using the 8-connected rule.

assigned an integer value of 3 (the integer value 2 has already been used to identify the object in the previous step). Figure 3.22a illustrates this pixel labeling in RGB space. Note that some perimeter pixels are missing; these have already been assigned to an object in the previous step. The result of reattaching the perimeter using the 8-connected rule is shown in Figure 3.22b. Perimeter pixels not assigned to an object are labeled as background.

Recall that the first step of the ICC algorithm was to apply a morphological closing algorithm to each object to remove all holes that were 1 pixel in size. However, there may still exist holes in the interior of objects that are larger than 1 pixel in size. The final step of the ICC algorithm is to apply a morphological closing operation to remove these holes. A flat, disk-shaped structuring element is used with radius 2. The final result of applying the ICC algorithm to the object in Figure 3.18a is shown in Figure 3.23a. The two component objects have clearly been separated. Figure 3.23b shows the outline of these two objects in the corresponding gray-scale image. Pseudocode for the ICC algorithm is given in Appendix A.



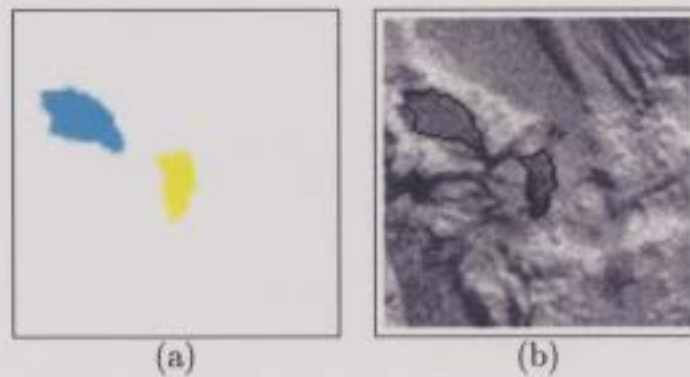


Figure 3.23: (a) Final result of applying the ICC algorithm to the object in Figure 3.18a. The two component objects have clearly been isolated. (b) Outline of separated objects in corresponding gray-scale image.

### 3.3.5 Remove Outliers

In order to remove outliers (i.e. data values that do not appear to be consistent with the rest of the data), all objects output from the ICC algorithm must undergo a final criteria test for area and length. The latter is defined as the length of the major axis of the ellipse that has the same normalized second central moments as the region [5]. All objects that have an area less than 50 or greater than 500, or a length greater than 50, are removed. These values were empirically chosen through the training process. In addition, objects that are located on the border of the  $100 \times 100$  pixel subimage are also removed.

The original gray-scale image in the ongoing example is shown in Figure 3.24a. The final result of applying the segmentation algorithm to this image is shown in RGB space in Figure 3.24b. The seal pup has been segmented, however several other non-seal pup objects have been segmented as well. It is the classifier's responsibility to identify each object as 'seal pup' or 'not seal pup'.

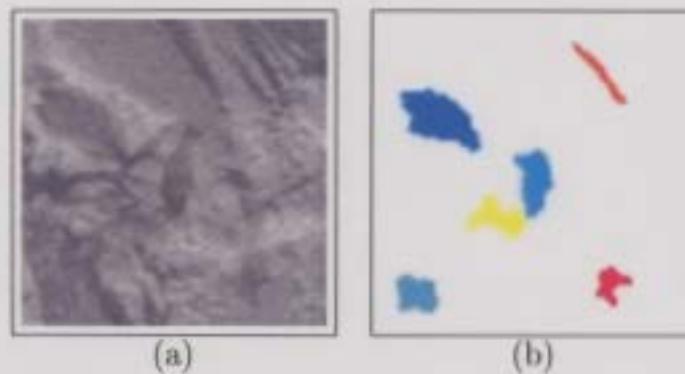


Figure 3.24: (a) Original  $100 \times 100$  pixel gray-scale image. (b) Final result of segmentation algorithm. The seal pup object has been segmented (located at centre of image), but several other non-seal pup objects (mostly areas of dark ice) have also been segmented.

### 3.4 Results

The following criteria were used to evaluate the performance of the segmentation algorithm on training data:

- **Excellent performance:** the algorithm cleanly segments 100% of the seal pup. No artifacts, such as shadows or ice, are attached to the segmented seal object.
- **Good performance:** the algorithm segments at least 90% of the seal pup. A very small part of the seal is segmented as background and/or small artifacts, such as shadows or ice, are segmented as part of the seal object; this slightly distorts the true shape of the seal.
- **Satisfactory performance:** the algorithm segments at least 50% of the seal pup. A portion of the seal object is segmented as background and/or artifacts, such as shadows or ice, are segmented as part of the seal object; the true shape of the seal is still apparent.
- **Poor performance:** the algorithm segments a very small portion of the seal pup ( $< 50\%$ ), or the seal is adjoined to other objects which are segmented

together as a single object; this does not accurately represent the true shape of the seal.

- **Failure:** the algorithm segments 0% of the seal pup.

In Section 3.2.3, several examples of challenges presented by the seal pup data are given (refer to images in Table 3.1). Table 3.3 demonstrates how the algorithm performed on these sample images. The first column of the table shows the original grayscale image, the second column shows the segmented objects in RGB space, and the third column contains remarks about the performance of the algorithm. Additional images showing algorithm performance are included in Appendix B.

Table 3.3: Performance of segmentation algorithm on sample images in Table 3.1.





Subimage	Segmentation Result	Performance of Algorithm
		<b>Excellent performance:</b> the seal pup object (shown in dark red) has been correctly segmented.
		<b>Good performance:</b> the seal pup object (shown in yellow) has been correctly segmented. A small part of the seal's shadow has also been segmented as part of the seal object.
Continued on next page		

Table 3.3 – continued from previous page













Subimage	Segmentation Result	Performance of Algorithm
		<b>Satisfactory performance:</b> the seal pup object (shown in yellow) has been partially segmented. Lighter gray level areas of the seal have been segmented as background.
		<b>Excellent performance:</b> the seal pup object (shown in red) has been correctly segmented.
		<b>Satisfactory performance:</b> the seal pup object (shown in yellow) has been partially segmented. Lighter gray level areas of the seal have been segmented as background.
		<b>Excellent performance:</b> the seal pup object (shown in yellow) has been correctly segmented.
		<b>Failure:</b> the algorithm failed to segment the seal pup which is partially occluded by adjacent ice.
Continued on next page		



Table 3.3 – continued from previous page

Subimage	Segmentation Result	Performance of Algorithm
		<p><b>Excellent performance:</b> the seal pup object (shown in light green) has been correctly segmented. However, several other non-seal pup objects have also been segmented from this complex background.</p>

Due to the complex nature of seal images, it is nearly impossible to develop an automated segmentation algorithm that will work perfectly for every image. As mentioned previously, the goal is to maximize the number of correctly segmented seal pups while eliminating as much background as possible. The segmentation algorithm was evaluated on 600 training images using the criteria described above. Table 3.4 presents the evaluation results. The algorithm performed excellent on over 48% of the training images. Approximately 45% of segmented seal pup objects were evaluated as good or satisfactory. In these cases, a few seal pup pixels may have been misidentified as background, or small artifacts, such as shadows or ice, may have been segmented as part of a seal. The algorithm's performance was unsatisfactory (evaluated as poor or failed performance) on 7% of the data. From the 600 training images, 1775 non-seal pup objects were also segmented. As discussed in Chapter 4, while these objects are not used to train the classifier, they are used to evaluate classifier performance.

It is assumed the algorithm will behave in a similar manner when applied to test data. When training the classifier to identify seal pups, it is necessary to include a valid representation of how seal objects appear when segmented; this includes seal

Table 3.4: Performance of segmentation algorithm on training data.

Performance Description	Total	% of total
Excellent	289	48.2%
Good	201	33.5%
Satisfactory	68	11.3%
Poor	31	5.2%
Failure	11	1.8%
<b>Total</b>	<b>600</b>	<b>100%</b>

objects that are perfectly segmented as well as those that have small imperfections (i.e. seal objects that have small artifacts adjoined to them or are slightly incomplete because small portions were misidentified as background). Therefore, training data evaluated as excellent, good, and satisfactory in Table 3.4 are used to train the classifier; this includes 93% of training data.

When applied to 300 test images (containing 300 seal pups), 1228 objects were segmented. These objects were analyzed by the classifier and identified as ‘seal pup’ or ‘not seal pup’ (as described in Chapter 4). Prior to automated classification, objects segmented from test data were manually reviewed and labeled in order to evaluate classifier performance and compute error rates. Through this process, it was observed that 297 seals and 931 outliers were segmented. Therefore, the algorithm failed to segment only 1% (3 seals) from test data. This 1% will be added to the classification error rate to compute the total error for the system.

### 3.5 Feature Selection

After the segmentation process, the resulting objects are represented and described in a form suitable for further computer processing. A segmented region may be *represented* in terms of its external characteristics (its boundary) or in terms of its internal

characteristics (the pixels comprising the region). A region is *described* based on the chosen representation using measurements called *features*. For example, a region may be represented by its boundary, and the boundary may be described by features such as its length or the orientation of the straight line joining its extreme points [68]. An external representation is chosen when the primary focus is on shape characteristics. An internal representation is selected when the primary focus is on regional properties, such as color and texture. It is often necessary to use both types of representation. It is of interest to select features that are simple to extract, invariant to irrelevant transformations, insensitive to noise, and useful for discriminating patterns in different categories. As discussed in Chapter 4, the choice and number of features affects classifier performance. In this thesis, the following nineteen features were chosen to represent segmented seal pup objects:

1. **Area** - the number of pixels in the region.
2. **Length** - the length (in pixels) of the major axis of the ellipse that has the same normalized second central moments as the region.
3. **Width** - the length (in pixels) of the minor axis of the ellipse that has the same normalized second central moments as the region.
4. **Elongation** - the ratio of the length,  $l$ , to width,  $w$ :  $l/w$ .
5. **Perimeter** - the number of pixels in the perimeter of the region. A pixel is part of the region perimeter if it is nonzero and it is connected to at least one zero-valued pixel.
6. **Compactness** - a measure of how close pixels are packed together. It is defined as  $p^2/a$ , where  $p$  is the perimeter and  $a$  is the area. Compactness is insensitive to uniform scale changes and is minimal for a disk-shaped region.

7. **Convex Area** - the number of pixels in the convex image of a region. The convex image is the smallest convex polygon that can contain the region, with all pixels within the polygon filled in (i.e. set to 'on').
8. **Convex Perimeter** - the length of the perimeter of the convex image of the region.
9. **Mean Pixel** - the average pixel intensity of the region:

$$\mu = \frac{1}{n} \sum_{i=1}^n r_i \quad (3.18)$$

where  $r_i$  = pixel intensity of the  $i^{th}$  pixel and  $n$  = number of region pixels.

10. **Pixel Standard Deviation** - the standard deviation of pixel intensities in the region:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (r_i - \mu)^2} \quad (3.19)$$

where  $r_i$  = pixel intensity of the  $i^{th}$  pixel,  $n$  = number of region pixels, and  $\mu$  = mean pixel value of the region.

11. **Minimum Pixel** - the minimum intensity of all pixels in the region:

$$\min_{1 \leq i \leq n} r_i \quad (3.20)$$

where  $r_i$  = pixel intensity of the  $i^{th}$  pixel and  $n$  = number of region pixels.

12. **Maximum Pixel** - the maximum intensity of all pixels in the region:

$$\max_{1 \leq i \leq n} r_i \quad (3.21)$$



where  $r_i$  = pixel intensity of the  $i^{th}$  pixel and  $n$  = number of region pixels.

13. **Moment Invariant 1** - this is defined as:

$$\phi_1 = \eta_{20} + \eta_{02} \quad (3.22)$$

where  $\eta_{pq}$  is the *normalized central moment* (refer to Appendix C for more details).

14. **Moment Invariant 2** - this is defined as:

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (3.23)$$

15. **Moment Invariant 3** - this is defined as:

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad (3.24)$$

16. **Moment Invariant 4** - this is defined as:

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (3.25)$$

17. **Moment Invariant 5** - this is defined as:

$$\begin{aligned} \phi_5 = & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (3.26)$$

18. **Moment Invariant 6** - this is defined as:

$$\phi_6 = (\eta_{20} - \eta_{02}) [(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (3.27)$$

19. **Moment Invariant 7** - this is defined as:

$$\begin{aligned}\phi_7 = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned}\quad (3.28)$$

An arrangement of features, such as those described above, is called a *pattern*. A very common pattern arrangement used in practice is a vector of the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad (3.29)$$

where each component,  $x_i$ , represents the  $i$ th feature and  $d$  is the total number of such features, or dimensionality, associated with the pattern [68]. All such feature vectors form a  $d$ -dimensional vector space  $\mathcal{X}$  called the *feature space*. In this thesis, a feature vector is formed for each segmented object using the nineteen features listed above. Thus, each object becomes a point in the nineteen-dimensional feature space. Furthermore, we presume that the *continuity assumption* of pattern recognition holds [77]. That is, two objects near in feature space should also resemble each other in real life. Therefore, it is assumed that objects are not randomly scattered in feature space, but that they exist in cloud-like distributions. When this continuity does not hold, it is unlikely that a classifier will learn well from a few example objects.

# Chapter 4

## Classification

This chapter describes the classification methods used to identify segmented objects as ‘seal pup’ or ‘not seal pup’. In the first section, a general classifier function is defined and the concept of one-class classification is introduced. A general overview of classifier design theory is given including a discussion of the many challenges associated with classifier training. In the next section, a summary of one-class classification theory is presented which includes error definitions, considerations for training one-class classifiers, techniques used to evaluate classifier performance, and a review of one-class methods. The next two sections give a detailed description of the two one-class methods applied in this research: the Parzen density estimation method and the Support Vector Data Description. The final section presents the methodology for optimizing and training these methods on seal data and compares their performance on test data.

## 4.1 Introduction

A classifier is a function that takes a set of features that characterize an object and uses them to determine the type, or *class*, of each object. Classifiers may give simple yes/no answers (e.g. ‘seal pup’ or ‘not seal pup’), or they may estimate the probability that an object belongs to each of the candidate classes. In many classification problems explicit rules do not exist to categorize an input object, but examples of objects from each defined class can be obtained. Therefore, a classifier can be constructed based on a finite set of pre-labeled training examples. To formalize this concept, let  $\mathcal{X}^{tr}$  represent a training set of  $N$  objects for which each object  $\mathbf{x}_i$  is assigned a label  $y_i$ . That is,

$$\mathcal{X}^{tr} = \{(\mathbf{x}_i, y_i) | i = 1, \dots, N\}. \quad (4.1)$$

For the classification, a function  $f(\mathbf{x})$  has to be inferred from the training set. This function is constructed such that for a given feature vector  $\mathbf{x}$  an estimate of the label is obtained,  $y = f(\mathbf{x})$ . Most often the type of classifier function  $f$  is chosen beforehand and just a few parameters of the function have to be determined. The function can be denoted by  $f(\mathbf{x}; \mathbf{w})$  to explicitly state the dependence on the parameters or weights  $\mathbf{w}$  [77].

In conventional multi-class classification problems, example patterns for two or more classes are available for training the classifier. An example of a simple two-class classification problem is illustrated in Figure 4.1. The training data consists of samples from two different species of seals: harp seals (represented by an asterisk, ‘\*’) and gray seals (represented by a plus sign, ‘+’). Each training object  $\mathbf{x}$  has two feature values, weight and length, and can therefore be represented as a point

in 2-dimensional feature space. A simple linear classifier  $f(\mathbf{x}; \mathbf{w})$  is modeled from the data set (represented by the straight line). From the two object features, the classifier estimates a label  $f(\mathbf{x}; \mathbf{w}) = +1$  (gray seal class) or  $f(\mathbf{x}; \mathbf{w}) = -1$  (harp seal class). The line  $f(\mathbf{x}; \mathbf{w}) = 0$  is the *decision boundary*. For most objects, it correctly estimates the labels. However, for two gray seals and three harp seals, a wrong label is assigned.

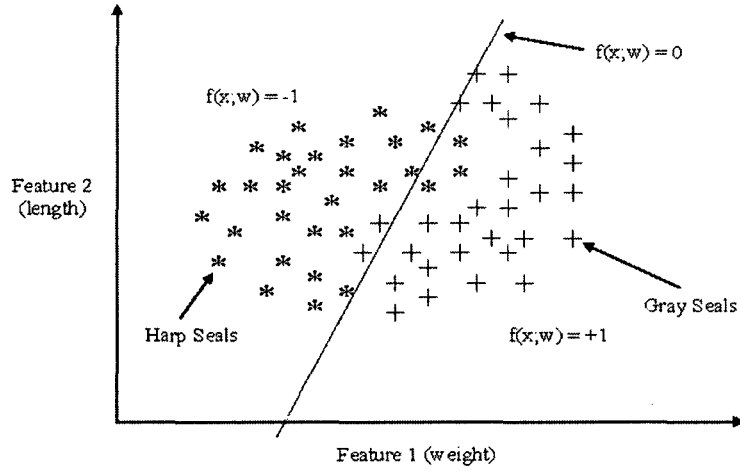


Figure 4.1: Scatterplot of a sample training set for gray seals and harp seals. A simple (linear) classifier is shown. From the two features, weight and length, the classifier estimates a label  $f(\mathbf{x}; \mathbf{w}) = +1$  (gray seal) or  $f(\mathbf{x}; \mathbf{w}) = -1$  (harp seal) for each object. The line  $f(\mathbf{x}; \mathbf{w}) = 0$  is the decision boundary.

In this thesis, a relatively novel classification approach called *one-class classification* is used. Unlike the two-class example give above, in one-class classification it is assumed that only information for *one* class, the target class, is available; information about all other objects, called outliers, is unavailable or ill-sampled. The boundary between the target and outlier classes has to be estimated from target data only. The task is to define this boundary such that it accepts as many target objects as possible while minimizing the chance of accepting outlier objects. Figure 4.2 illustrates a one-class classifier applied to the seal data from the two-class example. In this

case, the classifier is not trying to distinguish gray seals from harp seals. Instead, the classifier is used to describe the ‘seal’ class (target objects) and distinguish it from the ‘non-seal’ class (outliers).

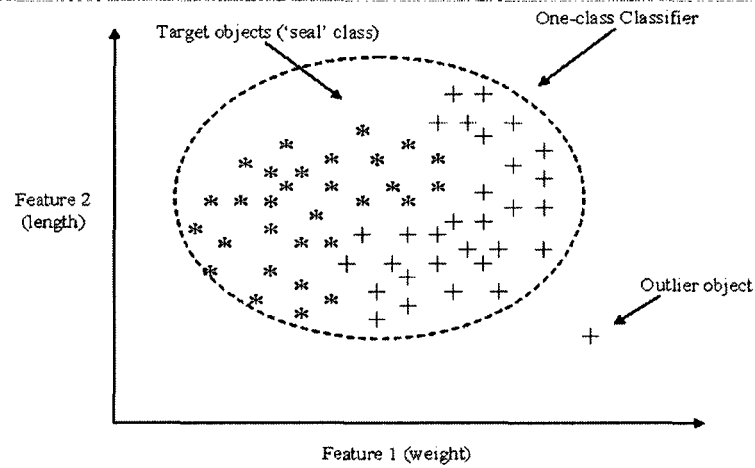


Figure 4.2: The boundary surrounding the data represents a one-class classifier used to identify seals (target objects). An object located outside this boundary would be classified as a non-seal (outlier object).

In general, the problem of one-class classification is harder than the problem of normal two-class classification. For normal classification the decision boundary is supported from both sides by examples of each of the classes. In the case of one-class classification only one set of data is easily available, so only one side of the boundary is covered. On the basis of one class it is difficult to decide how tight the boundary should fit around the data in each of the directions.

For the current research, a one-class classification approach is very appropriate. The target class consists of segmented seal pups for which ample training data is available. While some outlier data is available (sea ice, ocean, shadows, etc.), it is unknown whether these samples are well-representative of all possible non-seal pup objects that may be segmented. Therefore, these objects are grouped into the outlier class.

A more in-depth study of one-class classification is presented in Section 4.2. However, it is beneficial to first include a general overview of classifier design theory. Many of the concepts presented in the next section will be repeatedly considered in the discussion of one-class classification.

### 4.1.1 Classifier Design and Performance

The process of creating the optimal classifier for a given problem is quite complex. Many common issues arise in training a classifier including error minimization, generalization of the method, the curse of dimensionality, overfitting/underfitting, and the bias-variance dilemma. In this section, these concepts are discussed and suggestions are given on how to improve classifier performance.

#### Error Minimization

The parameters  $\mathbf{w}$  used to construct a classifier  $f(\mathbf{x}; \mathbf{w})$  may not produce the optimal classifier for the given problem. The optimal parameters  $\mathbf{w}^*$  of the function  $f$  are the parameters which give the smallest average error over *all possible* samples:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \mathcal{E}_{\text{true}}(f, \mathbf{w}, \mathcal{X}) \quad (4.2)$$

where the true error  $\mathcal{E}_{\text{true}}$  is defined as

$$\mathcal{E}_{\text{true}}(f, \mathbf{w}, \mathcal{X}) = \int \mathcal{E}(f(\mathbf{x}; \mathbf{w}), y) p(\mathbf{x}, y) d\mathbf{x} dy. \quad (4.3)$$

The integration in (4.3) is over the entire ‘true’ data distribution  $p(\mathbf{x}, y)$  (i.e. over the complete probability density of *all* possible objects  $\mathbf{x}$  and labels  $y$ ). This assumes that it is possible to define a probability density  $p(\mathbf{x}, y)$  in the complete feature space  $\mathcal{X}$ .

However, in almost all classification problems,  $p(\mathbf{x}, y)$  will be unknown. Therefore, an induction principle has to be used to approximate the true error [63, 77]. In practice, the true error is approximated by the empirical error on the training set:

$$\mathcal{E}_{\text{emp}}(f, \mathbf{w}, \mathcal{X}^{tr}) = \frac{1}{N} \sum_i \varepsilon(f(\mathbf{x}_i; \mathbf{w}), y_i) \quad (4.4)$$

where various error definitions are available for  $\varepsilon(f(\mathbf{x}_i; \mathbf{w}), y_i)$  including the 0-1-loss error, the mean squared error and the cross entropy error [77] (see Appendix D for details). By minimizing the empirical error  $\mathcal{E}_{\text{emp}}$  on the training data, it is expected that a set of weights  $\mathbf{w}$  will be found that produce a good classification.

## Generalization

Optimizing the classifier on the training data may be problematic if the set of training examples is an atypical set. That is, it may be unclear whether the training data distribution resembles the distribution in real life (i.e. the ‘true’ data distribution). In general, the larger the sample size, the more likely the true characteristics of the data can be determined. However, even if a large, characteristic data sample is available, the number of functions which approximates or precisely fits the data is quite large. Therefore, good classification of the training objects is not the main goal; rather good classification of *new and unseen* objects is the chief objective. This is called good *generalization*. To estimate how well a classifier generalizes, it should be tested with a new set of objects which has not been used for training. Using an independent test set avoids an overly optimistic estimate of classifier performance [77].



## Overfitting and Variance

When a function  $f(\mathbf{x}; \mathbf{w})$  minimizes the empirical error very well on a training set, but still shows a large true error  $\mathcal{E}_{\text{true}}$  on an independent test set, the function is said to be *overtrained* or *overfit*. An example of overfitting is shown in Figure 4.3. Here, a very flexible two-class classifier is trained on the harp seal/gray seal samples leading to a complex decision boundary. While such a decision may lead to perfect classification of the training samples (zero empirical error), it may lead to poor performance on future patterns (high true error).

When a very flexible function is trained on a new training sample  $\mathcal{X}^{tr}$  of the same size from the same distribution, a completely different solution for the weights  $\mathbf{w}$  will be obtained. This large *variance* in  $\mathbf{w}$  (and thus  $f(\mathbf{x}; \mathbf{w})$ ) over different training samples is undesirable.

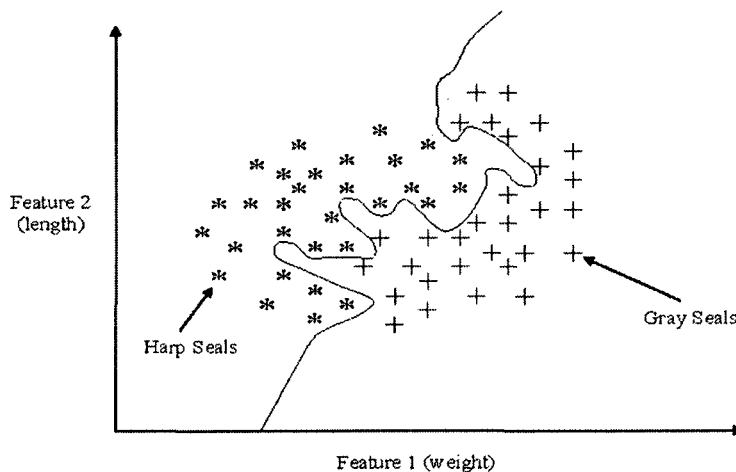


Figure 4.3: A heavily overtrained two-class classifier, trained on the harp seal/gray seal samples, produces a very complicated decision boundary. While such a decision may lead to perfect classification of the training samples (zero empirical error), it may lead to poor performance on future patterns (high true error).

## Curse of Dimensionality

The overfitting problem becomes worse when a large number of features per object is used. Since the function  $f(\mathbf{x}; \mathbf{w})$  should be defined for the complete feature space, the volume that should be described increases exponentially in the number of features. This is referred to as the *curse of dimensionality* [77, 78, 79]. One way to deal with the problems of overfitting and the curse of dimensionality is to reduce the number of features by selecting an appropriate subset of the existing features, or by combining the existing features in some way (for example, using Principal Component Analysis) [66, 80]. Another important quantity that may affect classifier performance is the number of training objects with respect to the dimensionality of the feature space. If the number of features is large, a small number of samples may be insufficient to estimate all free parameters  $\mathbf{w}$  in the function  $f(\mathbf{x}; \mathbf{w})$  with ample accuracy.

## Underfitting and Bias

Another problem that arises in classifier design is *underfitting*. In this case the function  $f(\mathbf{x}; \mathbf{w})$  is not flexible enough to follow all characteristics in the data. This type of classifier is said to have low complexity and shows a large *bias*. The bias measures how well the classifier fits the problem (high bias implies a poor fit) [66]. The linear classifier applied to the two-class problem in Figure 4.1 has a relatively low complexity. A more flexible non-linear classifier would most likely produce a better result.

## Bias-Variance Dilemma

A good fitting function for a given data sample should have both a small bias and a small variance. However, classifiers with increased flexibility to adapt to training

data (e.g. have more free parameters) tend to have lower bias but higher variance. Therefore, the best fitting function is a trade-off between the bias and variance contributions. This phenomenon is known as the *bias-variance dilemma* or *bias-variance trade-off* [77, 66]. A bias-variance trade-off between the classifiers in Figure 4.1 and Figure 4.3 is shown in Figure 4.4.

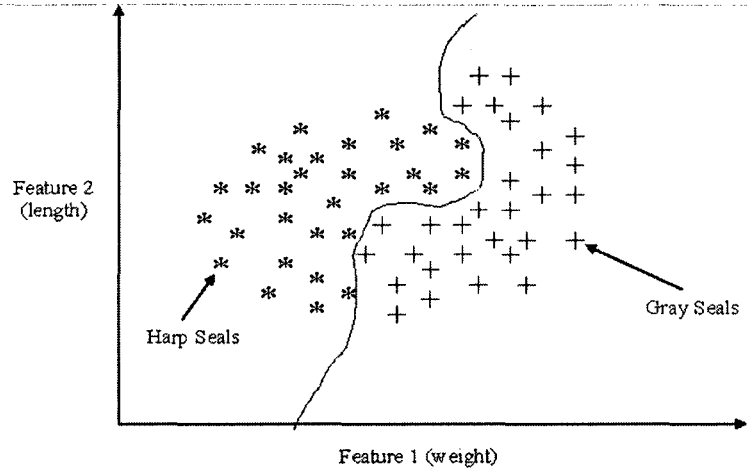


Figure 4.4: A bias-variance trade-off between the classifiers in Figure 4.1 and Figure 4.3. The classifier has a small bias and is robust against small changes in the training set (small variance).

## Structural Error

The bias-variance dilemma can be minimized by introducing prior knowledge into the design of  $f(\mathbf{x}; \mathbf{w})$ . By including prior knowledge, such as constraints on the form of  $f$  to the problem at hand, the complexity of the function is decreased while maintaining the flexibility [77]. Quite often, however, prior knowledge is not available or it is difficult to include the knowledge into the design of  $f(\mathbf{x}; \mathbf{w})$ . In this case, an extra error term  $\mathcal{E}_{\text{struct}}(f, \mathbf{w})$  is added to the empirical error. This structural error attempts to measure the complexity of the function  $f(\mathbf{x}; \mathbf{w})$ . Now the *total* error

must be minimized:

$$\mathcal{E}_{\text{tot}}(f, \mathbf{w}, \mathcal{X}^{tr}) = \mathcal{E}_{\text{emp}}(f, \mathbf{w}, \mathcal{X}^{tr}) + \lambda \mathcal{E}_{\text{struct}}(f, \mathbf{w}), \quad (4.5)$$

where  $\lambda$ , the *regularization parameter*, indicates the relative influence of the structural error with respect to the empirical error. If  $\lambda = 0$ , the structural error (and thus the complexity) of  $f(\mathbf{x}; \mathbf{w})$  will be ignored. If  $\lambda$  is large, a very simple function is obtained which completely ignores the data. This extended error is intended to produce a classifier with higher generalization. The structural error can be minimized by imposing smoothness constraints on the function  $f(\mathbf{x}; \mathbf{w})$ . Large fluctuations in the function are discouraged; the smoother the function, the lower the complexity. A number of approaches have been developed to approximate the structural error including counting the number of free parameters in the function, using weight decay, and considering the worse-case performance of the classifier [77]. A detailed discussion about this topic is given by Wolpert [81].

As the preceding discussion has shown, when designing a classifier, problems such as the generalization of the method, the curse of dimensionality, the bias-variance dilemma, and measuring the complexity of the solution must be considered. While numerous classification functions, errors, and optimization routines are available, the ‘true’ structure in the data is often quite difficult to model completely. It may be said that determining a suitable classifier for a given problem is more of an art than a science.

## 4.2 One-Class Classification

The term “one-class classification” was first coined by Moya *et al.* in 1993 [82]. A variety of other terms have been used in the literature to describe this classification approach including “data description” [77], “outlier detection” [83], “novelty detection” [84], and “concept learning” [85]. The terms one-class classification and data description will be used interchangeably in this thesis. Since the early 1990s, research in this area has grown considerably. Some recent applications of one-class classifiers include automated currency validation [86], diagnosis of interstitial disease in chest radiographs [87], image retrieval [88, 89], novelty detection in gene expression data [90], target recognition in SAR imagery [91], online signature verification [92], detecting masses in mammograms [93], bioacoustic monitoring [94], face recognition [95], and facial expression analysis [96].

The theory of one-class learning is covered extensively in the doctoral thesis of Tax [77]. The following is a summary of this theory including error definitions for one-class classifiers, considerations for training data descriptions, techniques used to evaluate classifier performance, and a review of one-class methods.

### 4.2.1 Error Definitions

In one-class classification, there are four possible situations of classifying an object; these are shown in Table 4.1. The fraction of target objects accepted by the classifier (true positives) is labeled  $f_{T+}$ . The fraction of targets rejected by the classifier (false negatives) is labeled  $f_{T-}$ ; this is called the error of the first kind,  $\mathcal{E}_I$ . The fraction of outlier objects rejected by the classifier (true negatives) is labeled  $f_{O-}$ . Finally, the fraction of outlier objects accepted by the classifier (false positives) is labeled  $f_{O+}$ ; this is called the error of the second kind,  $\mathcal{E}_{II}$ . In order to find a good one-class

classifier,  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  have to be minimized. The true positive and true negative objects do not contribute to the total error because they are classified correctly.

Table 4.1: The four possible situations of classifying an object in one-class classification.

	Object from target class	Object from outlier class
Classified as a target object	true positive, $f_{T+}$	false positive, $f_{O+}$ ( $\mathcal{E}_{II}$ )
Classified as an outlier object	false negative, $f_{T-}$ ( $\mathcal{E}_I$ )	true negative, $f_{O-}$

Recall that the complete probability density  $p(\mathbf{x}, y)$  is required to compute the true error  $\mathcal{E}_{\text{true}}$  (as defined in Equation (4.3)). In one-class classification, only the probability density of the target class  $p(\mathbf{x}|\omega_T)$  is known. Therefore, only the number of false negatives ( $\mathcal{E}_I$ ) can be minimized. The fraction false negative  $f_{T-}$  can be estimated using an independent test set drawn from the same target distribution or using cross-validation on the target training set. However, the fraction false positive  $f_{O+}$  is much harder to estimate. When no example outlier objects are available, this fraction cannot be estimated. However, minimizing just the fraction false negative will result in a classifier that labels *all* objects as target objects. In order to prevent this degenerate solution, outlier examples have to be available, or artificial outliers have to be generated. In the absence of examples, it is assumed that the outliers are drawn from a bounded uniform distribution around the target data and the description volume [77].

Using Bayes rule [66], the posterior probability for the target set can be computed

by

$$p(\omega_T|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_T)p(\omega_T)}{p(\mathbf{x}|\omega_T)p(\omega_T) + p(\mathbf{x}|\omega_O)p(\omega_O)} \quad (4.6)$$

Since the outlier distribution  $p(\mathbf{x}|\omega_O)$  is unknown, and the prior probabilities  $p(\omega_T)$  and  $p(\omega_O)$  are difficult to estimate, equation (4.6) cannot be used directly to compute the posterior probability for target objects. However, when it is assumed that  $p(\mathbf{x}|\omega_O)$  is independent of  $\mathbf{x}$  (i.e. it is uniformly distributed in the area of feature space under consideration),  $p(\mathbf{x}|\omega_T)$  can be used instead of  $p(\omega_T|\mathbf{x})$  [97].

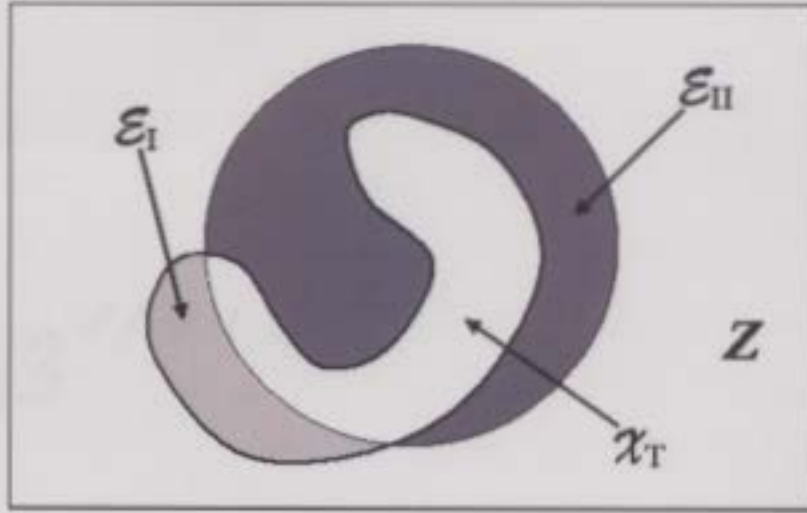


Figure 4.5: Regions in one-class classification. A spherically shaped one-class boundary is trained around a banana-shaped training set. The outliers are uniformly distributed in the rectangular block (as indicated by  $Z$ ). The light gray area represents the fraction of targets rejected,  $\mathcal{E}_I$ . The dark gray area represents the fraction of outliers accepted,  $\mathcal{E}_{II}$ .

Figure 4.5 illustrates the regions in one-class classification. The banana-shaped area is the target distribution  $\mathcal{X}_T$ . The circular boundary is the one-class classifier which should describe the data. In the absence of outliers examples, a uniform outlier distribution is assumed, indicated by  $Z$ . Note that some of the target data is

rejected (indicated by  $\mathcal{E}_I$ ) and some outliers are accepted (indicated by  $\mathcal{E}_{II}$ ). The most important feature of one-class classifiers is the tradeoff between the fraction of the target class that is accepted,  $f_{T+}$ , and the fraction of outliers that is rejected,  $f_{O-}$  (or equivalently, the tradeoff between  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$ ). It is evident from Figure 4.5 that this tradeoff cannot be avoided. Increasing the volume of the data description in order to decrease  $\mathcal{E}_I$  will automatically increase the number of accepted outliers, and therefore increase  $\mathcal{E}_{II}$ .

As Figure 4.5 also shows, using a uniform outlier distribution means that when  $\mathcal{E}_{II}$  is minimized, the data description with minimal volume is obtained. Therefore, instead of minimizing both  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$ , a combination of  $\mathcal{E}_I$  and the volume of the description can be minimized to obtain a good classifier [77]. However, when the true outlier distribution deviates from the uniform distribution, another data description will show better generalization performance. Of course, this cannot be checked without sample outliers. In this thesis, the 1775 non-seal pup objects segmented from training data will be used to estimate the outlier density and the fraction false positive,  $f_{O+}$ .

A structural error,  $\mathcal{E}_{\text{struct}}$ , may also be defined for one-class classifiers. Recall that for a conventional classifier, smoothness constraints on  $f(\mathbf{x}; \mathbf{w})$  can be imposed to reduce the structural error. For one-class classifiers not only should smoothness constraints be enforced, but also constraints on the closed boundary around the data. Unfortunately, these extra constraints make classifier design more difficult and amplify problems like the curse of dimensionality. As a result, one-class classification problems often require a larger sample size than conventional classification [77].



### 4.2.2 Distance/Resemblance Threshold

Two distinct elements are identified in all one-class classification methods:

1. A measure for the distance  $d(\mathbf{z}|\omega_T)$  or resemblance  $p(\mathbf{z}|\omega_T)$  of an object  $\mathbf{z}$  to the target class  $\omega_T$  (represented by the training set  $\mathcal{X}^{tr}$ );  $p(\mathbf{z}|\omega_T)$  (or  $d(\mathbf{z}|\omega_T)$ ) can also be interpreted as the chance of object  $\mathbf{z}$  given the target set  $\omega_T$ .
2. A threshold  $\theta$  on the distance  $d(\mathbf{z}|\omega_T)$  or resemblance  $p(\mathbf{z}|\omega_T)$ .

As discussed by Tax[77], new objects are accepted by the description when the distance to the target class is smaller than the threshold  $\theta_d$ :

$$f(\mathbf{z}) = I(d(\mathbf{z}|\omega_T) < \theta_d) \quad (4.7)$$

or when the resemblance is larger than the threshold  $\theta_p$ :

$$f(\mathbf{z}) = I(p(\mathbf{z}|\omega_T) > \theta_p) \quad (4.8)$$

where  $I(\cdot)$  is the indicator function defined as:

$$I(A) = \begin{cases} 1 & \text{if } A \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \quad (4.9)$$

One-class classification methods differ in their definition of  $p(\mathbf{z}|\omega_T)$  (or  $d(\mathbf{z}|\omega_T)$ ), and in their optimization of  $p(\mathbf{z}|\omega_T)$  (or  $d(\mathbf{z}|\omega_T)$ ) and thresholds with respect to  $\mathcal{X}^{tr}$ . Many one-class methods focus on optimizing the resemblance or distance model first and then optimize the threshold afterwards. However, a few methods optimize their model  $p(\mathbf{z}|\omega_T)$  or  $d(\mathbf{z}|\omega_T)$  to an *a priori* defined threshold [77]; different thresholds will give a different definition of  $p(\mathbf{z}|\omega_T)$  or  $d(\mathbf{z}|\omega_T)$ . In this thesis, the threshold is

derived directly from the training set and adjusted to accept a predefined fraction of the target class. For a target acceptance rate  $f_{T+}$ , the threshold  $\theta_{f_{T+}}$  is defined as

$$\theta_{f_{T+}} : \frac{1}{N} \sum_i I(p(\mathbf{x}_i|\omega_T) \geq \theta_{f_{T+}}) = f_{T+} \quad (\text{resemblance based method}) \quad (4.10)$$

or

$$\theta_{f_{T+}} : \frac{1}{N} \sum_i I(d(\mathbf{x}_i|\omega_T) \leq \theta_{f_{T+}}) = f_{T+} \quad (\text{distance based method}) \quad (4.11)$$

where  $\mathbf{x}_i \in \mathcal{X}^{tr}, \forall i$ .

### 4.2.3 ROC Curve

Using (4.10) (or (4.11)) we can, for varying  $f_{T+}$ , compute a threshold  $\theta_{f_{T+}}$  on the training set and then measure the resulting  $f_{O-}$  on a set of example outliers. When for all values of  $f_{T+}$ , the  $f_{O-}$  is measured, the Receiver Operating Characteristic (ROC) curve is obtained [98]. This curve shows how the fraction false positive changes for varying fraction false negative. The smaller these fractions are, the more this one-class classifier is preferred. Traditionally, the fraction true positive  $f_{T+}$  is plotted versus the fraction false positive  $f_{O+}$ , as shown in Figure 4.6. For this research, the 1775 sample outliers segmented from training data are used to measure the  $f_{O+}$  for varying  $f_{T+}$  on the target class.

While the ROC curve gives a good summary of the performance of a one-class classifier, it is difficult to compare two ROC curves. One way to summarize an ROC curve in a single number is to compute the Area Under the ROC Curve (AUC) [99]. To compute the AUC,  $\mathcal{E}_{\Pi}$  is integrated over varying thresholds (i.e. all possible errors

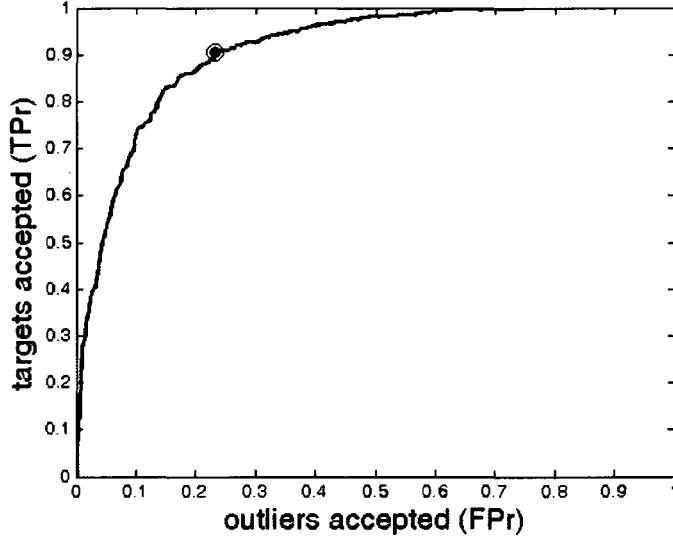


Figure 4.6: Example of a ROC curve. The fraction true positive  $f_{T+}$  is plotted versus the fraction false positive  $f_{O+}$ . The small dot on the curve represents the operating point.

of the first kind  $\mathcal{E}_I$ ). This gives the error

$$\mathcal{E}_M = \int_0^1 \mathcal{E}_{II}(\mathcal{E}_I) d\mathcal{E}_I = \int_0^1 \int_{\mathcal{Z}} I(p(\mathbf{z}) \geq \theta_f) d\mathbf{z} d\theta_f \quad (4.12)$$

where  $\theta_f$  is measured on the target set. This error measure does not evaluate a one-class classifier on the basis of one single threshold value, but integrates its performance over all threshold values. For the standard implementation of the AUC, smaller values indicate a better separation between target and outlier objects. However, for the implementation of the AUC included in the Data Description Toolbox[100] used in this thesis, larger AUC values indicate a better separation between target and outlier objects; this interpretation of the AUC is assumed for the remainder of the thesis.

Maximizing the AUC value does not guarantee optimal performance on test data; a trade-off between  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  must also be considered. It can therefore happen that for a specific threshold a one-class classifier with a lower AUC might be preferred over

another classifier with a higher AUC if, for that specific threshold, the fraction false positive ( $\mathcal{E}_{II}$ ) is smaller for the first classifier than the second one [100].

In some cases, a range of reasonable false negatives or false positives can be given. Therefore, the integration range for the AUC can be restricted to this specific range. This will result in a more honest comparison between different classifiers for the specific application [100]. Of course, for the actual application of a one-class classifier a specific threshold has to be chosen. This means that only one point of the ROC curve is used. An example of such an *operating point* is shown on the ROC curve in Figure 4.6.

#### 4.2.4 One-Class Methods

Several methods have been proposed to solve the one-class classification problem. Methods may be grouped into three main approaches: density estimation, boundary methods, and reconstruction models. The various one-class methods differ in their ability to exploit different characteristics of the data such as scaling of feature measurements, grouping of objects in clusters, and convexity of the data distribution and their placing in subspaces [77].

The most straight forward way to obtain a one-class classifier is to estimate the density of the training data [93] and to set a threshold on this density [77]. A variety of distributions may be assumed, such as a Gaussian or a Poisson distribution, and numerous discordancy tests are then available to test new objects [101]. Common density methods include the Gaussian model, the mixture of Gaussians, and the Parzen density model. The latter is described in detail in Section 4.3. Density methods work well when the sample size is sufficiently large and a flexible density model is used. If the sample size is not large enough, this approach may suffer from

the curse of dimensionality. Assuming a good probability model is used and the data size is sufficiently large, the density method approach has a big advantage: when one threshold value is optimized, a minimum volume is automatically found for the given probability density model. Therefore, only the high density areas of the target distribution are included and superfluous outliers will not be accepted [77].

In boundary methods, a closed boundary around the target set is optimized. Although the volume is not always actively minimized, most methods have a strong bias towards a minimal volume solution [77]. The size of the volume depends on the fit of the method to the data. In most cases, distances  $d$  to a set of objects in the training set are computed. Due to their focus on the boundary, the threshold on the output is straight forward to obtain. The number of objects required by boundary methods is smaller than is required for density methods. However, because the methods rely heavily on the distances between objects, they tend to be sensitive to the scaling of features. Therefore, well-defined distances must be considered when training these classifiers. Common boundary methods include the k-centers method, the nearest neighbor method (NN-d), and the Support Vector Data Description. The latter is described in detail in Section 4.4.

Reconstruction methods use prior knowledge about the data and make assumptions about the generating process to determine a model and then fit it to the data. New objects are then described in terms of a state of the generating model. These methods make assumptions about the clustering characteristics of the data or their distribution in subspaces. A set of prototypes or subspaces is defined and a reconstruction error is minimized. Examples include k-means clustering, learning vector quantization, self-organizing maps, Principal Component Analysis, diabolo networks and auto-encoder networks. These methods differ in their definition of the prototypes or subspaces, their reconstruction error, and their optimization routine. Reconstruc-

tion methods are not considered in this thesis, but more information can be found in [77].

### 4.3 Parzen Density Estimation

The most simple density model is the normal or Gaussian density [79]. For this model, the probability distribution for a  $d$ -dimensional object  $\mathbf{x}$  is given by:

$$p_{\mathcal{N}}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\} \quad (4.13)$$

where  $\boldsymbol{\mu}$  is the mean and  $\Sigma$  is the covariance matrix [66]. The Parzen density estimation [102] is an extension of the Gaussian model. The estimated density is a mixture of, most often, Gaussian kernels centered on the individual training objects, with a simplified diagonal covariance matrix  $\Sigma_i = h\mathcal{I}$ :

$$p_p(\mathbf{x}) = \frac{1}{N} \sum_i p_{\mathcal{N}}(\mathbf{x}; \mathbf{x}_i, h\mathcal{I}) \quad (4.14)$$

where  $\mathcal{I}$  is the identity matrix. The equal width  $h$  in each feature direction means that the Parzen density estimation assumes equally weighted features. As a result, this approach is sensitive to the scaling of features, especially if the sample size is small.

Training a Parzen density estimation consists of determining one single parameter, the width of the kernel  $h$ . This free parameter is typically optimized by maximizing the likelihood on the training data using leave-one-out [103, 104]. Producing a good description from this approach completely depends on how representative the training set is to the actual target distribution. While the computational cost for training the Parzen density estimation is quite low, the cost for testing can be expensive. All

training objects have to be stored and the distances to all training objects have to be calculated and sorted [77].

## 4.4 Support Vector Data Description

Support vector machines (SVM) have become a popular method in pattern classification and were originally developed for the discrimination of two-class classification problems [63, 105]. In support vector classifiers, the input vectors are mapped to a high dimensional feature space and then separated by the optimal linear hyperplane [106]. A variant of this method was proposed by Tax [77] in 2001 for one-class classification and has since been applied in a variety of research [89, 94, 96, 107]. Aptly called the Support Vector Data Description (SVDD), the one-class SVM attempts to describe the target class by finding a hypersphere boundary around the class with minimal volume. A better fit between the data boundary and the hypersphere model may be found by introducing a kernel function  $K$  which maps the data to a new feature space. This method is similar to unsupervised learning algorithms like Gaussian mixture models [108] or k-means clustering [109].

### 4.4.1 Derivation of SVDD

The SVDD is derived as a quadratic optimization problem. Let  $f(\mathbf{x}; \mathbf{w})$  represent a closed boundary, a hypersphere, around the target data. The hypersphere is characterized by a center  $\mathbf{a}$  and radius  $R$ . Assuming the hypersphere covers all training objects  $\mathcal{X}^{tr}$ , the empirical error is equal to 0. Analogous to the conventional SVM, the structural error is defined as

$$\mathcal{E}_{\text{struct}}(R, \mathbf{a}) = R^2 \tag{4.15}$$

which must be minimized with the constraints

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2, \quad \forall i. \quad (4.16)$$

To make the method more robust, the model should allow the possibility of outliers in the training set. In this case, the empirical error will not be 0. The total error now contains a structural and an empirical error contribution. Slack variables  $\xi, \xi_i \geq 0, \forall i$ , are introduced to enable soft boundary calculation and thus the minimization problem becomes

$$\mathcal{E}(R, \mathbf{a}, \xi) = R^2 + C \sum_i \xi_i \quad (4.17)$$

with constraints that almost all objects are within the hypersphere:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0, \quad \forall i. \quad (4.18)$$

The parameter  $C$  gives a tradeoff between the volume of the description and the errors. In Figure 4.7, a graphical representation of the hypersphere around some training data is given. It shows three objects on the boundary and one object  $\mathbf{x}_i$  that is rejected by the description.

The free parameters,  $\mathbf{a}$ ,  $R$ , and  $\xi$ , have to be optimized with respect to the constraints in (4.18). This can be accomplished by introducing Lagrange multipliers and constructing the Lagrangian [110]:

$$\begin{aligned} L(R, \mathbf{a}, \xi, \alpha, \gamma) = & R^2 + C \sum_i \xi_i \\ & - \sum_i \alpha_i \{ R^2 + \xi_i - (\mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{a} \cdot \mathbf{x}_i + \mathbf{a} \cdot \mathbf{a}) \} - \sum_i \gamma_i \xi_i \end{aligned} \quad (4.19)$$

with the Lagrange multipliers  $\alpha_i \geq 0$  and  $\gamma_i \geq 0$ , and where  $\mathbf{x}_i \cdot \mathbf{x}_j$  represents the



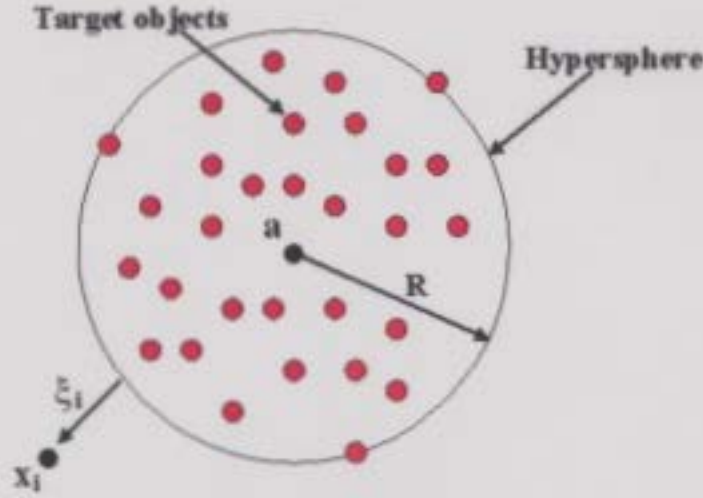


Figure 4.7: Graphical representation of a hypersphere with center  $\mathbf{a}$  and radius  $R$ . The sphere defines a boundary separating the target objects from the rest of the input space. Three objects are on the boundary, the support vectors. One object  $\mathbf{x}_i$  is outside the boundary and has  $\xi_i > 0$ .

inner product between  $\mathbf{x}_i$  and  $\mathbf{x}_j$ . For each object  $\mathbf{x}_i$ , a corresponding  $\alpha_i$  and  $\gamma_i$  are defined.  $L$  has to be minimized with respect to  $R$ ,  $\mathbf{a}$ , and  $\xi$ , and maximized with respect to  $\alpha$  and  $\gamma$ .

Using the Karush-Kuhn-Tucker theory [111], the partial derivatives are set to 0 producing the following constraints:

$$\frac{\partial L}{\partial R} = 0 : \quad \sum_i \alpha_i = 1 \quad (4.20)$$

$$\frac{\partial L}{\partial \mathbf{a}} = 0 : \quad \mathbf{a} = \frac{\sum_i \alpha_i \mathbf{x}_i}{\sum_i \alpha_i} = \sum_i \alpha_i \mathbf{x}_i \quad (4.21)$$

$$\frac{\partial L}{\partial \xi_i} = 0 : \quad \gamma_i = C - \alpha_i, \quad \forall i. \quad (4.22)$$

Constraint (4.22) can be rewritten as  $\alpha_i = C - \gamma_i$ . Instead of the constraints that  $\gamma_i \geq 0$  and  $\gamma_i = C - \alpha_i$ , a new constraint on  $\alpha_i$  can be introduced:

$$0 \leq \alpha_i \leq C, \quad \forall i. \quad (4.23)$$

Provided (4.23) is satisfied, the Lagrange multipliers  $\gamma_i$  can be computed by  $\gamma_i = C - \alpha_i$  and  $\gamma_i \geq 0$  automatically holds. Assuming all constraints hold, (4.19) can be rewritten as

$$\begin{aligned} L(R, \mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) &= R^2 - \sum_i \alpha_i R^2 + C \sum_i \xi_i - \sum_i \alpha_i \xi_i - \sum_i \gamma_i \xi_i \\ &\quad + \sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x}_i - 2 \sum_i \alpha_i \mathbf{a} \cdot \mathbf{x}_i + \sum_i \alpha_i \mathbf{a} \cdot \mathbf{a} \end{aligned} \quad (4.24)$$

$$\begin{aligned} &= 0 + 0 + \sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x}_i - 2 \sum_i \alpha_i \sum_j \alpha_j \mathbf{x}_j \cdot \mathbf{x}_i + 1 \cdot \sum_{i,j} \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &= \sum_i \alpha_i \mathbf{x}_i \cdot \mathbf{x}_i - \sum_{i,j} \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \end{aligned} \quad (4.25)$$

The minimization of the error function (4.25) with constraints (4.23) is a well-known quadratic programming problem and standard algorithms exist to solve this [77].

When an object  $\mathbf{x}_i$  is *within* the hypersphere, the inequality in constraint (4.18) is satisfied and the corresponding Lagrange multiplier becomes zero:  $\alpha_i = 0$ . For objects satisfying the equality  $\|\mathbf{x}_i - \mathbf{a}\|^2 = R^2 + \xi_i$ , the object is located at or outside the boundary and the corresponding Lagrange multiplier becomes positive:  $\alpha_i > 0$ . When an object obtains  $\alpha_i = C$ , the object is regarded as an outlier and will not be accepted by the data description (i.e. the hypersphere description is not adjusted further to include the corresponding object  $\mathbf{x}_i$  in the description).

In equation (4.21), the center of the sphere  $\mathbf{a}$  is expressed as a linear combination of objects with weights  $\alpha_i$ . For the computation of  $\mathbf{a}$ , objects with 0 weight ( $\alpha_i = 0$ ) can be disregarded; only objects with positive weight ( $\alpha_i > 0$ ) are required. In the minimization of (4.25), often a large fraction of the weights become 0. Therefore, the sum in (4.21) is usually over a small fraction of objects  $\mathbf{x}_i$  with  $\alpha_i > 0$ . Only these objects, called the *support vectors*, are required for the description of the data set.

To test if a new object  $\mathbf{z}$  is accepted by the description, the distance from the

object to the center of the hypersphere  $\mathbf{a}$  has to be calculated. The object is accepted when this distance is smaller than or equal to the radius:

$$\|\mathbf{z} - \mathbf{a}\|^2 = (\mathbf{z} \cdot \mathbf{z}) - 2 \sum_i \alpha_i (\mathbf{z} \cdot \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \leq R^2, \quad (4.26)$$

where  $R^2$  can be determined by calculating the squared distance from the center  $\mathbf{a}$  to any support vector  $\mathbf{x}_k$  on the boundary:

$$R^2 = (\mathbf{x}_k \cdot \mathbf{x}_k) - 2 \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_k) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j). \quad (4.27)$$

The SVDD can now be written as

$$\begin{aligned} f_{SVDD}(\mathbf{z}; \boldsymbol{\alpha}, R) &= I(\|\mathbf{z} - \mathbf{a}\|^2 \leq R^2) \\ &= I\left((\mathbf{z} \cdot \mathbf{z}) - 2 \sum_i \alpha_i (\mathbf{z} \cdot \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \leq R^2\right) \end{aligned} \quad (4.28)$$

where  $I$  is the indicator function, as defined in (4.9). A bonus feature of the SVDD is that it offers a direct estimate of the error it makes on the target set [77]. The fraction of the target objects which become support vectors is an estimate of the fraction of target objects rejected by the description.

When objects which should be rejected (negative examples) are available, they can be used during training to improve the SVDD (i.e. to obtain a tighter boundary around the data in the areas where outlier objects are present). In contrast with the target examples, which should be within the hypersphere, the negative examples should be outside it. Formulation for an SVDD that uses negative examples to improve the description boundary is given in Appendix D.

#### 4.4.2 The Kernel Trick

The hypersphere is a very rigid model of the boundary of the data. In general, this model will not fit the data well. An improved target data description can be achieved using the so-called *kernel trick*, similar to that used in the nonlinear SVM [105]. The general idea is to map data vectors from the input space to a high-dimensional kernel space using an implicit nonlinear mapping  $\Phi$  and then minimize the volume of the hypersphere containing the data in the kernel space. To illustrate this idea, assume we are given a mapping  $\Phi$  which improves the fit:

$$\mathbf{x}^* = \Phi(\mathbf{x}) \quad (4.29)$$

Applying this mapping to (4.25) and (4.28) gives:

$$L = \sum_i \alpha_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (4.30)$$

and

$$f_{SVDD}(\mathbf{z}; \boldsymbol{\alpha}, R) = I \left( \Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}) - 2 \sum_i \alpha_i \Phi(\mathbf{z}) \cdot \Phi(\mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \leq R^2 \right) \quad (4.31)$$

Since all mappings  $\Phi(\mathbf{x})$  in (4.30) and (4.31) occur only in inner products with other mappings, a new *kernel function* of two input variables can be defined:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \quad (4.32)$$

Replacing all occurrences of  $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$  in (4.30) and (4.31) by this kernel gives:

$$L = \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (4.33)$$

and

$$f_{SVDD}(\mathbf{z}; \boldsymbol{\alpha}, R) = I \left( K(\mathbf{z}, \mathbf{z}) - 2 \sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \leq R^2 \right) \quad (4.34)$$

The kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$  is called a Mercer kernel because it can be written as an inner product of two functions [105, 112]. In this formulation, the mapping  $\Phi$  is never used explicitly; it is only defined implicitly by the kernel  $K$ . The advantage of the kernel trick is that the introduction of the kernels does not introduce much extra computational costs. The optimization problem remains identical in the number of free parameters [77].

Several kernel functions have been proposed for the support vector classifier [113]. However, it appears that not all kernels that were proposed for the SVC can be used by the SVDD. In most cases the data classes are elongated, which is useful for discrimination between two classes, but is harmful for one-class classification [114]. The two most common kernels used with the SVDD are the polynomial kernel and the Gaussian kernel. The polynomial kernel is given by

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^p \quad (4.35)$$

where the free parameter  $p$  gives the degree of the polynomial kernel. As argued by Vapnik [63], this kernel maps the objects into the high dimensional feature space by adding products of the original features, up to degree  $p$ . For example, a 2D vector

$(x_1, x_2)$  is mapped to  $(x_1, x_2, x_1x_2, x_1^2, x_2^2)$  when a polynomial kernel with  $p = 2$  is used [115].

The Gaussian kernel is given by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{s^2}\right) \quad (4.36)$$

where the free parameter  $s$  is the width of the kernel. For small values of  $s$  the SVDD resembles a Parzen density estimation, while for large  $s$  the original hypersphere solution is obtained [115]. Tax [77] investigates the behaviour of these kernel functions in detail. He concludes that the polynomial kernel suffers from the large influence of the norms of the object vectors. While the Gaussian kernel results in a tighter description than the polynomial kernel, it requires more data to support the more flexible boundary. Both the polynomial and Gaussian kernels are investigated in this thesis.

## 4.5 Experimental Results

This section describes the experimental results of applying the Parzen density estimation and SVDD classifiers to segmented seal data. First, a review of feature optimization techniques is given. This is followed by a description of the general methodology followed for training and testing the classifiers including parameter optimization. Finally, the results of applying the data descriptions to test data are given and classifier performance is evaluated.

### **4.5.1 Feature Optimization**

In order to have a good distinction between target and outlier objects, good representation of the data is essential. The performance of both the Parzen density estimation and the SVDD critically depends on the scaling of data and is often harmed by data distributions in subspaces [114]. Therefore, three different approaches have been applied to optimize data features prior to training the classifier; scaling of the data, Principal Component Analysis, and kernel whitening. The latter two methods use feature reduction, while scaling maintains the original dimension of the feature space. Classifier performance is evaluated on pre-processed data and compared to classification results on data which is not pre-processed. It is expected that feature optimization will improve classifier performance.

#### **Scaling Data**

For the first feature optimization approach, a scaling map is computed on the target data such that its mean is shifted to the origin and the variances of all features are scaled to one. Test data is pre-processed with the mapping that is computed for the training set.

#### **Principal Component Analysis**

The second feature optimization approach is Principal Component Analysis (PCA). The central idea of PCA is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate

(the first principal component), the second greatest variance on the second coordinate, on so forth. PCA then eliminates the components that contribute the least to the variation in the data set. In this thesis, the principal components that contribute less than 1% to the total variation in the training data are eliminated. This constraint results in 6 principal components in the transformed subspace. Test data is pre-processed with the transformation matrix that is computed for the training set. For a detailed description of PCA, please refer to Jolliffe [116] and Jackson [117].

### **Kernel Whitening**

The final feature optimization approach is kernel whitening. In a study conducted by Schölkopf *et al.* [118], it was indicated that nonlinear principal components from kernel PCA had better recognition rates than the corresponding number of linear principal components from linear PCA. In addition, the performance for nonlinear components can be further improved by using more components than possible in the linear case. Kernel whitening uses the idea of kernel PCA to extract the nonlinear principal features of the data set. After mapping the data to this new feature space (implicitly defined by the kernel function), feature directions with (almost) zero variance are removed and the other features are rescaled to unit variance. By the kernel PCA and rescaling, the resulting data has zero mean with an identity covariance matrix [114]. In principal, this data can now be described by *any* one-class classifier. Mathematical details of the kernel whitening transformation are given in Appendix D.

The efficiency of mapping the data to the new representation with unit variance depends on the choice of the kernel and parameters. A polynomial or Gaussian kernel is typically chosen. A suitable kernel may be selected by using the Chernoff distance



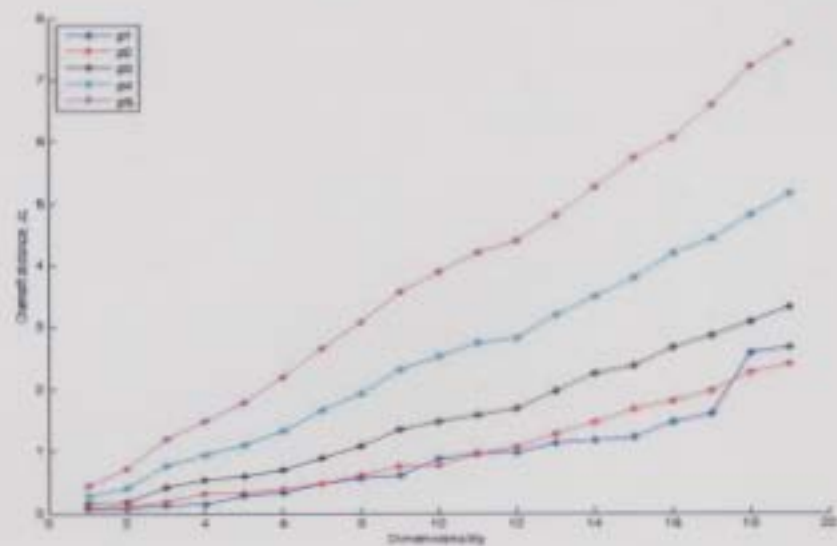
between the target and outlier data [114]:

$$J_C = \frac{1}{2} \log \left[ \frac{|\mathcal{I}/2 + \Sigma_O/2|}{|\Sigma_O|^{1/2}} \right] \quad (4.37)$$

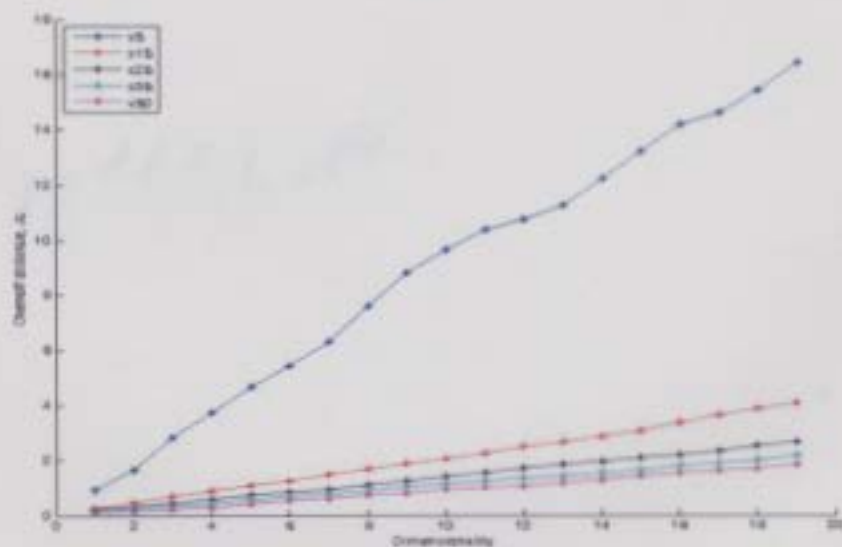
where  $\Sigma_O$  is the covariance matrix of the outlier class and  $\mathcal{I}$  is the identity matrix (see Appendix D for details). This distance is relatively cheap to compute, and thus the expensive optimization of the classifier and the computation of the AUC for all combinations of kernel definition and data dimensionality can be avoided. For each kernel definition a near optimal dimensionality can be estimated. Then the classifier and the AUC have to be computed and compared for only these combinations of kernel and dimensionality.

In an attempt to determine the optimal kernel, the Chernoff distance is computed for the training data using a polynomial kernel with the degree ranging from  $p = 1$  to  $p = 5$  and using a Gaussian kernel with the width parameters  $s = 5, 15, 25, 35, 50$ . For each of the kernels, the number of used principal components  $d'$  (dimensionality) is varied. In Figure 4.8a, the Chernoff distance between the target and outlier classes is shown for the polynomial kernel. This plot implies that the higher the degree, the faster the distance increases. This suggests that using higher degrees and higher dimensionalities gives the best performance. However, in practice very good performances have been obtained with lower values of  $p$  and  $d'$  [114]. In Figure 4.8b, the Chernoff distance between the target and outlier classes is shown for the Gaussian kernel. This plot suggests that the smaller the kernel width, the faster the distance increases. Unfortunately, there is no obvious choice of kernel from these plots.

Since a suitable kernel is difficult to select from the plots in Figure 4.8, the data is pre-processed by each kernel using different parameters and varying data dimensionalities, and then the Parzen density estimation and SVDD are trained on the data.



(a)



(b)

Figure 4.8: The Chernoff distance for varying data dimensionalities of the training data which is pre-processed using (a) the polynomial kernel with degrees  $p = 1, \dots, 5$ , and (b) the Gaussian kernel with width parameters  $s = 5, 15, 25, 35, 50$ .

For each combination of parameter and dimensionality, the classifier is optimized by maximizing the AUC. This AUC value is then plotted as a function of the retained dimensionality of the kernel PCA. Figure 4.9 displays the AUC values computed for the Parzen density estimation when the data is pre-processed using the polynomial kernel. A similar plot was produced for the Gaussian kernel, but the best results were clearly obtained with the polynomial kernel. As the plot in Figure 4.9 shows, a polynomial kernel of degree 1 with 13 retained principal components gives the best result. Therefore, this configuration is used to pre-process data for the Parzen classifier. It is interesting to note that this result is counterintuitive from the Chernoff distance results in Figure 4.8a which suggest that using higher degrees and higher dimensionalities for the polynomial kernel gives the best performance.

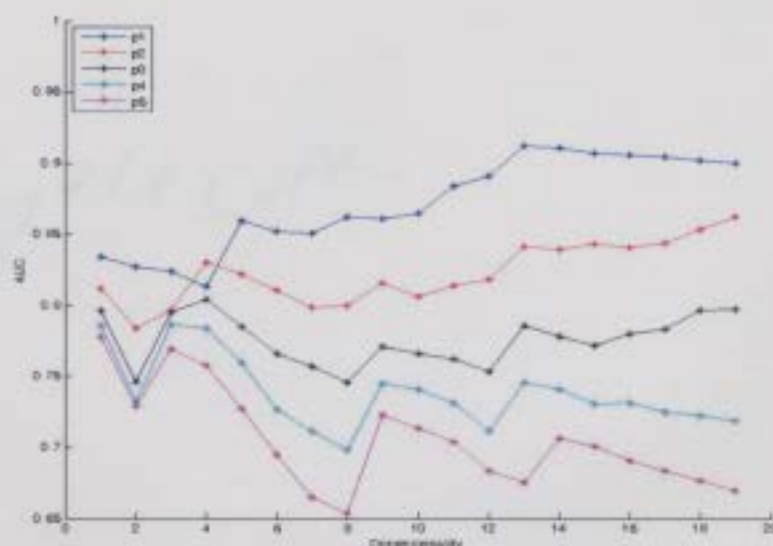
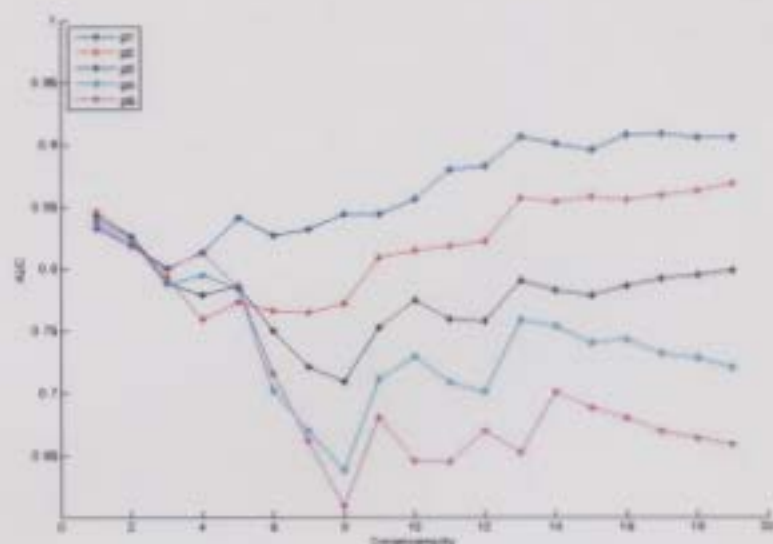
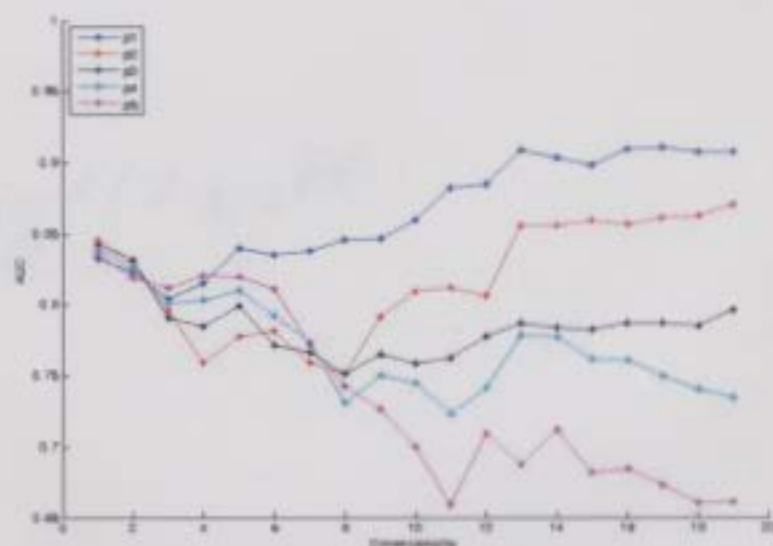


Figure 4.9: For the polynomial kernel with degrees  $p = 1, \dots, 5$ , and for varying dimensionality  $d'$ , a Parzen density estimation is optimized by maximizing the AUC. This AUC value is plotted as a function of  $d'$ . For this classifier, a kernel of degree 1 with 13 retained principal components gives the best result.

Figure 4.10 displays the AUC values computed for the SVDD when the data is pre-processed using the polynomial kernel. Two different SVDD configurations are



(a)



(b)

Figure 4.10: For the polynomial kernel with degrees  $p = 1, \dots, 5$ , and for varying dimensionality  $d'$ , a SVDD is optimized by maximizing the AUC. This AUC value is plotted as a function of  $d'$ . In (a) a polynomial kernel of degree 1 is used to construct the SVDD. In (b) a Gaussian kernel with  $s = 10$  is used to construct the SVDD. In both cases,  $p = 1$  with 17 retained principal components gives the best result.

tested; in Figure 4.10a the SVDD is constructed with a polynomial kernel of degree 1 and in Figure 4.10b the SVDD is constructed with a Gaussian kernel using  $s = 10$  (the SVDD kernel should not be confused with the kernel function used in the whitening process). As both plots show, a polynomial kernel of degree 1 with 17 retained principal components gives the best result for both support vector data descriptions. Therefore, this configuration is used to pre-process data for the SVDD. Similar plots were produced for the Gaussian kernel, but results were not as good when compared to the polynomial kernel.

### 4.5.2 Methodology for Training and Testing

This research uses the Data Description Toolbox[100] and PRTools4[119] to create and apply one-class classifiers. After the data has been pre-processed, as described in the previous section, the next step in training a classifier is to determine the optimal free parameter(s) by maximizing the AUC on training data. This optimization process is conducted for the Parzen method, for the SVDD with a polynomial kernel, and for the SVDD with a Gaussian kernel.

For the Parzen density estimation there is only one free parameter to optimize, the kernel width  $h$ . This parameter is typically optimized by maximizing the likelihood on the training data using leave-one-out [103, 104]. However, a different approach is taken here. First, a reasonable range of values for  $h$  is determined based on the scaling of the data. This range will change depending on the pre-processing method applied. For each value  $h$  in the specified range, the average AUC value is computed using 5-fold cross validation. That is, the training data is randomly split into 5 groups of equal or near-equal size. For each group, the ROC curve is plotted and the corresponding AUC value is computed. The AUC for all 5 groups is averaged

and recorded as the AUC for the given parameter  $h$ . This process is repeated for all parameter values in the given range. The parameter that produces the maximum AUC is chosen as the optimal kernel width for the given data set.

For the SVDD, the free parameters are the Lagrange multipliers  $\alpha$ . From these Lagrange multipliers, the center  $\mathbf{a}$  and the threshold value  $R$  can be computed. Thus, the number of free parameters is  $N$ , the size of the target set. However, for the SVDD implementation in the Data Description Toolbox, the user only has to supply the number of false negatives  $f_{T-}$  (which is varied for computation of the ROC curve) and the kernel parameter(s) from which all other free parameters are determined. For a polynomial kernel there is one free parameter, the degree  $p$  of the polynomial. For a Gaussian kernel there is also one free parameter, the width  $s$ . The process for determining the optimal parameter for the SVDD kernel is analogous to the process used for the Parzen classifier. First, a reasonable range of values for the parameter is given based on the type of kernel. For a polynomial kernel,  $p = 1, \dots, 5$  is used. For a Gaussian kernel,  $s = 1, \dots, 50$  is tested. For each parameter value in the specified range, the average AUC value is computed using 5-fold cross validation as described above. The parameter that produces the maximum AUC is chosen as the optimal kernel parameter for the given data set.

The next step in the training process is to select a specific operating point. As discussed in Section 4.2.2, the threshold on the distance  $d$  or resemblance  $p$  is derived directly from the training set and adjusted to accept a predefined fraction of the target class. For a target acceptance rate  $f_{T+}$ , the threshold  $\theta_{f_{T+}}$  is defined by Equation (4.10) (or (4.11)). In this thesis, the target acceptance rate on the training data is determined from the ROC plot produced from the optimal parameter(s). For training the classifier, the fraction false negative should be no more than 10% (i.e. minimum target acceptance rate of 90%). Therefore, for all operating points on the ROC curve

for which  $\mathcal{E}_I \leq 0.1^1$ , the combination of  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  that gives the minimum total value is selected as the operating point for training the classifier. In other words, the operating point must satisfy the condition

$$\min_{0 \leq \mathcal{E}_I \leq 0.1} (\mathcal{E}_I + \mathcal{E}_{II}) \quad (4.38)$$

For example, consider the ROC curve in Figure 4.11. This curve is produced when the Parzen density estimation is applied to training data that is pre-processed using PCA ( $h = 2.439$  is the optimal kernel width). The condition in (4.38) is satisfied by the operating point  $(\mathcal{E}_I, \mathcal{E}_{II}) = (0.09625, 0.2406)$ , as identified by the dot on the curve.

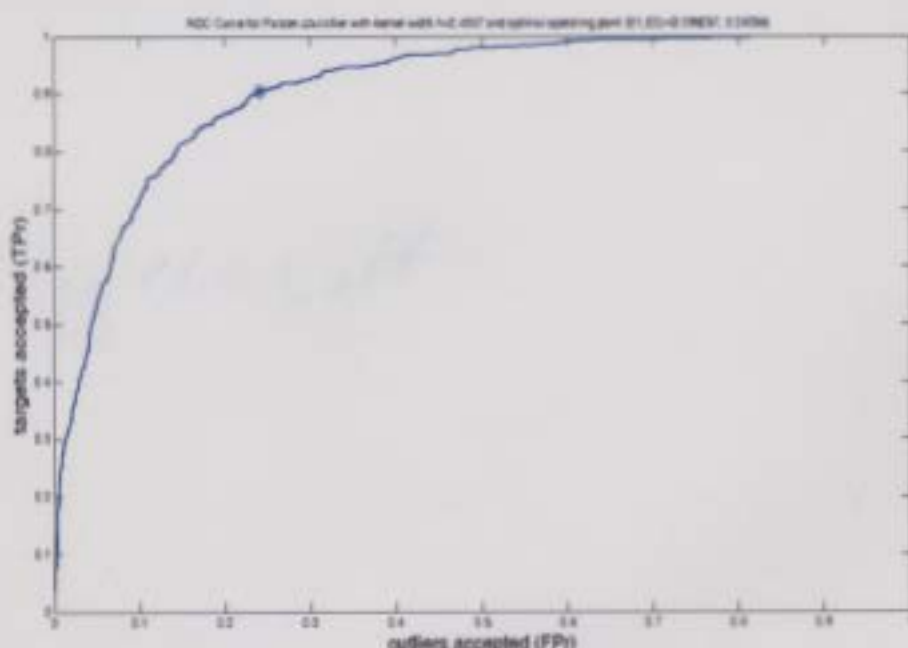


Figure 4.11: ROC curve for the Parzen density estimation trained on seal pup data with  $h = 2.439$  and operating point  $(\mathcal{E}_I, \mathcal{E}_{II}) = (0.09625, 0.2406)$ .

The optimal parameter(s) and operating point determined in the previous steps

<sup>1</sup>The ROC curve does not explicitly show the fraction false negative. Rather,  $\mathcal{E}_I$  is obtained from the ROC by computing  $1 - f_{T+}$ .

are now used to train the classifier. Finally, the data description is applied to test data and its performance is evaluated and compared to other classifiers. The process for training and testing each classifier, as described above, is summarized as follows:

1. Optimize features (scaling of data, PCA, kernel whitening).
2. Optimize classifier parameter(s) by maximizing AUC using 5-fold cross validation on training data.
3. Using the ROC curve produced from the optimal parameter(s), find  $\mathcal{E}_I$  that satisfies the condition in (4.38). This is considered the optimal operating point.
4. Use the optimal parameter(s) and operating point in the previous steps to construct the classifier.
5. Apply the classifier to test data and evaluate its performance.

### 4.5.3 Final Results

Table 4.2 shows the final classification results for the objects segmented from test data. The first column defines the pre-processing method applied to the data. The second column gives the optimal parameter value for the given data description. The third and fourth columns give the values of  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$ , respectively, for the classifier performance on test data. The fifth column gives the total percentage of successful classifications (defined as  $(f_{T+} + f_{O-})/2$ ) and the final column gives the AUC produced from the optimal parameter(s) on training data. For each of the three classification methods (Parzen, SVDD with polynomial kernel, SVDD with Gaussian kernel), the optimal values for  $\mathcal{E}_I$ ,  $\mathcal{E}_{II}$ , the percentage of successful classifications, and the AUC, are highlighted in **bold** typeface. It is important to note that  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  are equally



Table 4.2: Final classification results on test data.

Method	Parameter	$\mathcal{E}_I$	$\mathcal{E}_{II}$	$(f_{T+} + f_{O-})/2$	AUC
<b>Parzen density estimation</b>					
no pre-processing	h=75.610	10.77%	37.72%	75.76%	0.8462
scaled	h=2.624	11.79%	<b>25.54%</b>	81.34%	0.9093
PCA	h=2.439	10.77%	25.75%	<b>81.74%</b>	0.9095
kernel whitening ( $p = 1, d' = 13$ )	h=13.628	<b>8.75%</b>	46.88%	72.19%	<b>0.9119</b>
<b>SVDD with polynomial kernel</b>					
no pre-processing	p=0.5	9.43%	39.01%	75.78%	0.8146
scaled	p=1	<b>9.09%</b>	34.05%	78.43%	0.8698
PCA	p=1	9.43%	34.38%	78.10%	0.8694
kernel whitening ( $p = 1, d' = 17$ )	p=1	15.49%	<b>26.72%</b>	<b>78.90%</b>	<b>0.9080</b>
<b>SVDD with Gaussian kernel</b>					
no pre-processing	s=48	<b>9.42%</b>	66.27%	62.16%	0.7067
scaled	s=7	11.79%	31.03%	78.59%	0.8858
PCA	s=7	12.80%	31.03%	78.09%	0.8808
kernel whitening ( $p = 1, d' = 17$ )	s=12	15.49%	<b>27.16%</b>	<b>78.68%</b>	<b>0.9093</b>

weighted when analyzing classifier results (i.e. it is equally important to minimize both the number of false negatives and the number of false positives).

As discussed in Chapter 3, objects segmented from test data were manually reviewed and labeled in order to compute error rates. Through this process, it was observed that 297 seals and 931 outliers were segmented (a total of 1228 objects). The percentages for  $\mathcal{E}_I$  in Table 4.2 were computed for the 297 target objects. The percentages for  $\mathcal{E}_{II}$  were computed for the 931 outlier samples.

### Parzen Density Estimation

For the Parzen method, the percentage of successful classifications ranges from 72.19% for data pre-processed with kernel whitening, to 81.74% for data pre-processed with PCA. As expected, applying the Parzen classifier with no pre-processing does not

produce the optimal result as features are not equally weighted. The large differences in magnitude between feature types in the original input space is evident by the large width parameter,  $h = 75.610$ .

When predicting classifier performance using AUC values, it is expected that the best separation between targets and outliers will be produced when the data is pre-processed via kernel whitening. As Table 4.2 shows, the kernel whitening approach does result in the smallest  $\mathcal{E}_I$  value (8.75%). However, the corresponding  $\mathcal{E}_{II}$  value (and thus the total error) from this configuration is significantly high (46.88%). A low  $\mathcal{E}_I$  and high  $\mathcal{E}_{II}$  may result from underfitting the classifier. Therefore, the kernel whitening approach is considered suboptimal for the Parzen method.

The smallest value for  $\mathcal{E}_{II}$  is obtained when the data is pre-processed via scaling (25.54%), closely followed by the  $\mathcal{E}_{II}$  value obtained when the data is pre-processed with PCA (25.75%). Since a smaller  $\mathcal{E}_I$  value and smaller total error is obtained from the PCA configuration, this is considered the optimal solution for the Parzen density estimation. Using this data description (PCA + Parzen), approximately 82% of test data is correctly classified.

## SVDD

For the SVDD, the polynomial kernel and Gaussian kernel produce similar results. The percentage of successful classifications ranges from 62.16% (Gaussian kernel with no pre-processing) to 78.90% (polynomial kernel with kernel whitening). For the SVDD constructed with a polynomial kernel, optimizing the classifier resulted in a kernel of degree  $p = 1$  for all pre-processing methods. When the data is not pre-processed, the optimal polynomial kernel has degree  $p = 0.5$ . For the SVDD constructed with a Gaussian kernel, optimizing the classifier resulted in a range of width parameters for the various pre-processing methods ( $s = 7$  for scaling and PCA,

$s = 12$  for kernel whitening). When the data is not pre-processed, the optimal Gaussian kernel width is significantly larger ( $s = 48$ ).

For the polynomial kernel, the smallest  $\mathcal{E}_I$  value is produced when the data is pre-processed via scaling (9.09%). However, when predicting classifier performance using AUC values, it is expected that the best separation between targets and outliers will be produced when the data is pre-processed via kernel whitening. While this approach produces the largest  $\mathcal{E}_I$  value (15.49%), it also results in the smallest  $\mathcal{E}_{II}$  value (26.72%) and greatest number of successful classifications (78.90%). Since the optimal balance between  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  is produced via kernel whitening, this configuration is considered the optimal SVDD using a polynomial kernel.

For the Gaussian kernel, the smallest  $\mathcal{E}_I$  value is surprisingly produced when the data is not pre-processed (9.42%). However, the corresponding  $\mathcal{E}_{II}$  value is significantly higher (66.27%). When predicting classifier performance using AUC values, it is again expected that the best separation between targets and outliers will be produced when the data is pre-processed via kernel whitening. Analogous to the polynomial kernel, this approach also produces the largest  $\mathcal{E}_I$  value with the Gaussian kernel (15.49%). However, since the smallest  $\mathcal{E}_{II}$  value (27.16%) and greatest number of successful classifications (78.68%) are also produced with kernel whitening, this approach is selected as the optimal choice for the SVDD with a Gaussian kernel.

The SVDD with kernel whitening produces a similar result for the polynomial and Gaussian kernels; in both cases, close to 79% of test data is correctly classified. Both kernels produce the same  $\mathcal{E}_I$  value (15.49%), but the polynomial kernel produces a slighter lower  $\mathcal{E}_{II}$  value (26.72%). Therefore, this configuration is chosen as the optimal classifier for the SVDD approach. For both kernels, pre-processing the data via scaling produces the next best results with 78.43% (polynomial kernel) and 78.59% (Gaussian kernel) of test data correctly classified.

## Optimal Classifier

In summary, pre-processing the data with PCA and applying the optimized Parzen density estimation produces the best results on test data (81.74% success rate). The Parzen density estimation with scaled data produces the next best result (81.34% success rate), followed by the SVDD with polynomial kernel ( $p = 1$ ) and pre-processing via kernel whitening (78.90% success rate). Density methods produce the best data descriptions when the target distributions for the training and test data are (approximately) equal [77]. Therefore, it can be concluded that the distribution of seal objects in the training set and test set are very similar. However, even for the optimal data description, over 25% of outlier objects are classified as targets. Therefore, the target and outlier distributions still overlap to a moderate degree.

As discussed in Chapter 3, the segmentation algorithm failed to extract 1% of seals from test data. That is, for 3 out of 300 test images, 0% of the seal object was segmented. This error is added to the total error rate  $((\mathcal{E}_I + \mathcal{E}_{II})/2)$  for the optimal classifier (PCA + Parzen density estimation) to produce the total error rate for the optimal system (segmentation + classification). As Table 4.3 shows, the total success rate for the optimal system is 80.74%.

Table 4.3: Total error rate for the optimal system; this includes the segmentation error plus the error rate for the optimal classifier (PCA + Parzen).

Segmentation error rate	1%
Classification error rate	18.26%
Total error rate	19.26%
Total success rate	80.74%

# Chapter 5

## Conclusions and Recommendations

### 5.1 Summary of Research Methods

In this thesis, a method is developed to automatically recognize harp seal pups in black-and-white aerial photographs captured by DFO. It is expected that an automated approach will conserve time, reduce sources of error, and save money for estimating total pup counts. The algorithm uses image segmentation and pattern recognition methods to extract objects from aerial images and then classify these objects as either ‘seal pup’ or ‘not seal pup’.

Each cropped subimage is first enhanced by applying contrast stretching to increase the dynamic range of pixel values. The foreground is segmented from the background by applying an adaptive thresholding algorithm that divides the subimage into smaller regions and then uses between-class variance and histogram skewness to compute an optimal threshold for each subdivided region. The binary threshold image is “cleaned” by using Canny edge detection to further define object boundaries, line dissection to disconnect adjacent objects, and removal of objects based on size constraints. The Isolate Connected Components algorithm further separates adjacent

objects with minimal distortion to object shape. Finally, outlier objects are removed based on area and length measurements.

Nineteen feature measurements are computed for each segmented object. Three different approaches have been applied to optimize data features prior to training the classifier; scaling of the data, PCA, and kernel whitening. The latter two methods use feature reduction, while scaling maintains the original dimension of the feature space. For PCA, the principal components that contribute less than 1% to the total variation in the training data are eliminated. This constraint results in 6 principal components in the transformed subspace. For kernel whitening, a polynomial kernel of degree 1 is used to transform the data and the number of principal components depends on the classifier; 13 and 17 principal components are selected for the Parzen density estimation and SVDD, respectively.

Two one-class classifiers are trained and tested on seal pup data: the Parzen density estimation and the SVDD. For the SVDD, a polynomial and Gaussian kernel are tested and compared. For each classifier, the optimal parameter(s) are computed by maximizing the AUC using 5-fold cross validation on training data. Using the ROC curve produced from the optimal parameter(s), the operating point is chosen by selecting  $\mathcal{E}_I$  that satisfies  $\min_{0 \leq \mathcal{E}_I \leq 0.1} (\mathcal{E}_I + \mathcal{E}_{II})$ . The optimal parameter(s) and operating point are then used to train the data description. Finally, classifier performance is evaluated on objects segmented from test data.

## 5.2 Summary of Results and Conclusions

Automated recognition of harp seal pups in digitized aerial photographs is a non-trivial task. Aerial images of whelping concentrations may contain a variety of features including open waterways, sea ice, shadows, adult seals, seal pups and other marine

life. Sea ice is varied in size, shape, and texture; it may appear flat and smooth or rough and jagged. All of these features present challenges when attempting to distinguish seal pups from their surroundings.

The segmentation algorithm must be robust to complex conditions such as uneven illumination, shadows, occlusions, and objects grouped together. One reoccurring issue with the seal images is that whitecoats and small ice chunks may be similarly shaped. Another problem is that seal pup pixels and the adjacent background pixels sometimes share similar gray level values and as a result, these areas are segmented together as a single object. To address these problems, the segmentation algorithm incorporates robust procedures such as the adaptive thresholding algorithm and the original Isolate Connected Components algorithm which attempts to separate individual seal objects with minimal distortion to object shape. Even with the variety of challenges presented, the segmentation algorithm performs good on the majority of images. Algorithm performance was evaluated as excellent, good, or satisfactory on 93% of seals segmented from 600 images used for training. When applied to 300 test images, the segmentation algorithm successfully segments 297 seals; therefore, the algorithm failed to segment only 1% of seals from test data. Of course, the segmentation algorithm also segments a large number of non-seal pup objects (outliers); 1775 outliers were segmented from training data and 931 outliers were segmented from test data.

While a large number of outliers were segmented from seal pup data, it is difficult to know how representative these objects are of the true outlier distribution. Conversely, objects from the target class (seal pups) are well-sampled. For this situation, a one-class classifier is preferred over a conventional classifier. A one-class classifier is used to describe the target class and then detect new data that is characteristic (seal pups) or uncharacteristic (outliers) of the target data. Therefore, a well-sampled

class of outliers is not required.

Classifier performance is affected by how well the training data represents the true data distribution, the size of the sample, the type and number of object features, and the complexity of the model. The Parzen density estimation and the SVDD are both sensitive to the scaling of data and its distribution in subspaces. For both classifiers, pre-processing the data using scaling and PCA significantly improves results over data that is not pre-processed. Pre-processing the data using kernel whitening improves performance for the SVDD but not for the Parzen density estimation. The success of kernel whitening largely depends on choosing a suitable kernel function and parameter values.

The ROC curve provides a good way to evaluate classifier performance and choose an optimal operating point. While the AUC provides a single measurement to compare classifiers, maximizing its value does not guarantee optimal performance on test data; a trade-off between  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  must also be considered. It can therefore happen that for a specific threshold a one-class classifier with a lower AUC might be preferred over another classifier with a higher AUC if, for that specific threshold, the fraction false positive ( $\mathcal{E}_{II}$ ) is smaller for the first classifier than the second one. A combination of AUC,  $\mathcal{E}_I$ , and  $\mathcal{E}_{II}$  values are considered in this thesis to determine the optimal classifier.

The Parzen density estimation and the SVDD both produce satisfactory results on test data. As Table 4.2 shows, the fraction of false negatives is quite reasonable, ranging from 8.75% to 15.49%. However, the fraction of false positives has a much broader range and can be fairly high, ranging from 25.54% to 66.27%. The total number of successful classifications ranges from 62.16% to 81.74%.

The largest AUC value and the smallest  $\mathcal{E}_I$  value for the Parzen method are obtained when the data is pre-processed using kernel whitening. However, the resulting



$\mathcal{E}_{II}$  (and thus the total error) from this configuration is significantly high (46.88%). Therefore, this approach is considered suboptimal. The smallest value for  $\mathcal{E}_{II}$  is obtained when the data is pre-processed via scaling (25.54%), closely followed by the  $\mathcal{E}_{II}$  value obtained when the data is pre-processed with PCA (25.75%). Since a smaller  $\mathcal{E}_I$  value and smaller total error is obtained from the latter configuration, this is considered the optimal solution for the Parzen density estimation. Using this data description (PCA + Parzen), 81.74% of test data is correctly classified.

For the SVDD, the polynomial and Gaussian kernels produce similar results. For both kernels, the smallest  $\mathcal{E}_{II}$  values, smallest total error, and largest AUC values are obtained when the data is pre-processed using kernel whitening. However, both configurations produce the largest  $\mathcal{E}_I$  values (15.49%). While all other configurations produce a smaller  $\mathcal{E}_I$ , the kernel whitening approach is still considered optimal because it produces the largest number of successful classifications (approximately 79% for both kernels). Since the polynomial kernel produces a slightly smaller  $\mathcal{E}_{II}$  than the Gaussian kernel, this configuration is considered optimal for the SVDD approach.

In summary, pre-processing the data with PCA and applying the optimized Parzen density estimation produces the best results on test data (81.74% success rate). The Parzen density estimation with scaled data produces the next best result (81.34% success rate), followed by the SVDD with polynomial kernel ( $p = 1$ ) and pre-processing via kernel whitening (78.90% success rate). Recall that the segmentation algorithm failed to extract 1% of seals from test data. Adding this error to the classification error rate for the Parzen density estimation (18.26%) produces the total error rate for the optimal system (19.26% error rate = 80.74% success rate).

Density methods produce the best data descriptions when the target distributions for the training and test data are (approximately) equal. Therefore, it can be concluded that the distribution of seal objects in the training set and test set are very

similar. However, even for the optimal data description, over 25% of outlier objects are classified as targets. Therefore, the outlier distribution overlaps the target distribution to a moderate degree. Unfortunately, creating a tighter data description to minimize the number of outliers accepted will also increase the number of targets rejected. Improving the segmentation algorithm may produce a better representative sample of target objects, minimize the number of outliers segmented, and thus reduce the number of outliers accepted as targets. In addition, a better separation between target and outlier distributions may be obtained by another one-class classification method (such as other boundary or reconstruction methods).

The algorithm developed in this thesis for the detection and classification of harp seal pups produces promising first results. While this research is just one possible solution, the development of an automated pattern recognition system for counting seal pups certainly appears to be feasible. If such a system could significantly reduce the time and costs for producing population estimates, then it is worth investing in further research to improve the accuracy and robustness of the segmentation and classification methods.

## 5.3 Recommendations

The following are recommendations for future work.

1. **Improve segmentation algorithm** - While the segmentation algorithm has a high success rate for extracting seal pups, it also segments a large number of other, unwanted objects. Improving the adaptive thresholding algorithm and the techniques that separate adjacent and occluded seals may minimize the number of outliers segmented. Template matching may also be considered as

a segmentation approach<sup>1</sup>. In this case, a finite set of templates that represent the variety of seal pup shapes would need to be created.

2. **Test other one-class classifiers** - In a literature search of one-class classification applications, the Parzen density estimation and SVDD often produce the best results. However, there are over 20 one-class classifiers currently available. It would be of interest to investigate how other one-class classifiers perform on the seal data. Alternative approaches to consider include other boundary methods (k-centers, nearest neighbor (NN-d)) and reconstruction methods (k-means clustering, learning vector quantization, self-organizing maps, diabolo networks and auto-encoder networks).
3. **Combine one-class classifiers** - As in any type of classification, one classifier hardly ever captures all characteristics of the data. To improve classifier performance and increase robustness of the classification, the results of different classifiers (which may differ in complexity or training algorithm) could be combined [77, 97]. Classifiers can be combined in several ways. One approach is to use different feature sets and combine the classifiers trained on each set. Another approach is to train several different classifiers on one feature set and combine these. The effectiveness of combining one-class classifiers has been researched by a number of people in recent years [97, 120, 121, 90], and it may be worth investigating for the problem studied in this research.
4. **Incorporate weights into the total error** - The most important feature of one-class classifiers is the tradeoff between  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$ . Consider two one-class classifiers with the following error values:

---

<sup>1</sup>The idea of template matching is to create a model of an object of interest (the template, or kernel) and then to search over the image of interest for objects that match the template.

Classifier 1  $\rightarrow \mathcal{E}_I = 0.11, \mathcal{E}_{II} = 0.44$ , total error = 0.55

Classifier 2  $\rightarrow \mathcal{E}_I = 0.16, \mathcal{E}_{II} = 0.38$ , total error = 0.54

Classifier 2 has a smaller total error (0.54), but only by a small margin. For the given problem however, it may be more important to maximize the number of true positives (i.e. minimize  $\mathcal{E}_I$ ). To reflect the greater importance placed on the value of  $\mathcal{E}_I$ , both error values could be weighted (i.e.  $\alpha\mathcal{E}_I + \beta\mathcal{E}_{II}$ , where  $\alpha$  and  $\beta$  are the weights for  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$ , respectively). Assume that it is twice as important to minimize  $\mathcal{E}_I$  over  $\mathcal{E}_{II}$ . Setting  $\alpha = 2$  gives the following:

Classifier 1  $\rightarrow$  total error =  $2(0.11) + 0.44 = 0.66$

Classifier 2  $\rightarrow$  total error =  $2(0.16) + 0.38 = 0.70$

Now classifier 1 produces the smallest total error. Incorporating weights into the total error may assist in the choosing the optimal classifier for a particular problem. For this research, it is assumed that  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  are equally weighted. However, if the pattern recognition tools developed in this thesis were incorporated into a software system, the option to weight  $\mathcal{E}_I$  and  $\mathcal{E}_{II}$  differently may be important to the end-user.

5. **Compare manual and automated methods on new data** - It would be of interest to compare automated detection and classification results to traditional manual results on new data. An automated approach would only be practical (and preferred) if the results were similar to manual detection within a certain error margin. A small total error rate may be acceptable for the automated approach (when compared to the manual process) if the time and cost savings are substantial.

6. **Test data from other surveys** - Large format negatives from aerial surveys must be digitized by a third party company and this process is quite expensive. Therefore, only the aerial photographs from the 1999 survey were digitized and provided for this research. However, since 1990, DFO have conducted aerial surveys of whelping concentrations at four to five year intervals. Therefore, an abundance of data from other surveys is available and could be digitized to further test the segmentation and classification algorithms developed herein. Empirically chosen parameters for the segmentation algorithm (e.g. size constraints) may need to be adjusted for other survey data if aerial photographs were captured at a different altitude. In addition, if the distribution of target data from other surveys is different from the target distribution of data collected in 1999, a new classifier will need to be trained.
7. **Complete system design** - Once the pattern recognition methods have been studied further and adjusted for optimal performance, they may be incorporated into a complete software system for estimating pup production of harp seals. Input to this system would be an original large image (as shown in Figure 2.4) and the output would be the number and locations of seals in the image. In addition, it may be possible to include a module for on-line learning; this would allow end-users to train the classifier on new data sets. It is important to note that a software system would most likely be an aid for counting seals, not a user replacement. Manual user quality control would still have to be incorporated into the system to confirm automated detections.
8. **Other survey methods** - More and more often, environmental monitoring problems are turning to the computational world for automated solutions. It is highly likely that an automated approach for estimating harp seal pup produc-

tion will be established in the coming years; the exact form of that approach is yet to be determined. While this research focused on the current survey method of using black-and-white aerial photography, another survey method may lend itself to more robust pattern recognition tools. The feasibility of multi-spectral scanning, ultraviolet photography, and infrared photography may be considered.

# Bibliography

- [1] M. O. Hammill and G. Stenson, "Abundance of northwest atlantic harp seals (1960-2005)," DFO Canadian Science Advisory Secretariat Res. Doc. 2005/090, 2005.
- [2] M. O. Hammill and G. B. Stenson, "Estimated consumption of atlantic cod (*gadus morhua*) and some other prey by grey seals (*halichoerus grypus*) and harp seals (*phoca groenlandica*), in the southern gulf of st. lawrence (nato division 4t)," DFO Canadian Science Advisory Secretariat Res. Doc. 2003/067, 2002.
- [3] B. P. Healey and G. B. Stenson, "Estimating pup production and population size of the northwest atlantic harp seal (*phoca groenlandica*)," DFO Canadian Stock Assessment Secretariat Res. Doc. 2000/081, 2000.
- [4] G. B. Stenson, "Computer-assisted image analysis techniques to support marine mammal population assessments," DFO Science Strategic Fund Project Application Form, 2001.
- [5] *MATLAB®*, ©1994-2006 The MathWorks, Inc., [www.mathworks.com](http://www.mathworks.com).
- [6] G. B. Stenson, M. O. Hammill, J. Lawson, J. F. Gosselin, and T. Haug, "2004 pup production of harp seals, *pagophilus groenlandicus*, in the northwest atlantic," DFO Canadian Science Advisory Secretariat Res. Doc. 2005/037, 2005.
- [7] "Underwater world: The harp seal," Fisheries and Oceans Canada online resources ([www.dfo-mpo.gc.ca](http://www.dfo-mpo.gc.ca)).
- [8] DFO, "Northwest atlantic harp seals," DFO Science Stock Status Report E1-01, 2000.
- [9] M. O. Hammill and G. Stenson, "Application of the precautionary approach and conservation reference points to the management of atlantic seals: A discussion paper," DFO Canadian Science Advisory Secretariat Res. Doc. 2003/067, 2003.
- [10] W. D. Bowen, "Role of marine mammals in aquatic ecosystems," *Mar. Ecol. Prog. Ser.*, vol. 158, pp. 267–274, 1997.

- [11] D. E. Sergeant, "Calculation of harp seal production in the western north atlantic," *Int. Comm. Northwest Atl. Fish. Redbook 1971, Part III, Res. Doc. 71/7*, pp. 157–183, 1971.
- [12] ———, "Estimating numbers of harp seals," *Rapp. P.-v. Reun. Cons. Int. Explor. Mer*, vol. 169, pp. 274–280, 1975.
- [13] T. Benjaminsen and T. Oritsland, "The survival of year-classes and estimates of production and sustainable yield of northwest atlantic harp seals," *Int. Comm. Northwest Atl. Fish. Res. Doc. 75/121*, 1975.
- [14] G. H. Winters, "Production, mortality, and sustainable yield of northwest atlantic harp seals (*pagophilus groenlandicus*)," *J. Fish. Res. Board. Can.*, vol. 35, pp. 1249–1261, 1978.
- [15] J. G. Cooke, "Population estimates of northwest atlantic harp seal (*phoca groenlandica*) based on age structure data," *Can. J. Fish. Aquat. Sci.*, vol. 42, pp. 468–473, 1985.
- [16] D. M. Lavigne, S. Innes, W. Barchard, and W. G. Doubleday, "The 1977 census of northwest atlantic harp seals, *pagophilus groenlandicus*," *Int. Comm. Northwest. Atl. Fish. Sel. Pap.*, vol. 6, pp. 55–70, 1980.
- [17] D. M. Lavigne, S. Innes, K. Kalpakis, and K. Ronald, *An aerial census of western Atlantic harp seals (Pagophilus groenlandicus) using ultraviolet photography*, *Int. Comm. Northwest Atl. Fish. Res. Doc. 75/144, Sr. 3717*, 1982.
- [18] W. D. Bowen and D. E. Sergeant, "Mark-recapture estimates of harp seal pup (*phoca groenlandica*) production in the northwest atlantic," *Can. J. Fish. Aquat. Sci.*, vol. 40, no. 6, pp. 728–742, 1983.
- [19] ———, "A mark-recapture estimate of 1983 harp seal pup production in the northwest atlantic," *Northwest Atl. Fish. Organ. SCR Doc. 85/I/1, Ser., N935*, 1985.
- [20] G. B. Stenson, M. O. Hammill, J. F. Gosselin, and B. Sjare, "1999 pup production of harp seals, *phoca groenlandica*, in the northwest atlantic," *DFO Canadian Stock Assessment Secretariat Res. Doc. 2000/080*, 2000.
- [21] G. B. Stenson, R. A. Myers, M. O. Hammill, I.-H. Ni, W. G. Warren, and M. C. S. Kingsley, "Pup production of harp seals *phoca groenlandica*, in the northwest atlantic," *Can. J. Fish. Aquat. Sci.*, vol. 50, pp. 2429–2439, 1993.
- [22] G. B. Stenson, L.-P. Rivest, M. O. Hammill, J. F. Gosselin, and B. Sjare, "Estimating pup production of harp seals, *pagophilus groenlandicus*, in the northwest atlantic," *Marine Mammal Science*, vol. 19, no. 1, pp. 141–160, January 2003.



- [23] K. J. Gaston and M. A. O'Neill, "Automated species identification: why not?" *Philosophical Trans. Royal Soc. Long. B*, vol. 359, pp. 655–667, 2004.
- [24] T. Moyo, S. Bangay, and G. Foster, "The identification of mammalian species through the classification of hair patterns using image pattern recognition," *In Proc. AFRIGRAPH 2006, Cape Town, South Africa*, pp. 177–181, 2006.
- [25] S. Sen, S. Narasimhan, and A. Konar, *A Novel Algorithm for Automatic Species Identification Using Principal Component Analysis*. Springer Berlin / Heidelberg, 2005, vol. 3776, pp. 605–610.
- [26] E. Vilches, I. A. Escobar, E. E. Vallejo, and C. E. Taylor, "Data mining applied to acoustic bird species recognition," *18th International Conference on Pattern Recognition (ICPR'06)*, pp. 400–403, 2006.
- [27] D. Chesmore, "Automated bioacoustic identification of species," *An. Acad. Bras. Cience*, vol. 76, no. 2, pp. 435–440, 2004.
- [28] P. Somervu and A. Harma, *Bird song recognition based on syllable pair histograms*, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04), Montreal, Canada, 2004.
- [29] H. P. Jeffries, M. S. Berman, A. D. Poularikas, C. Katsinis, I. Melas, K. Sherman, and L. Bivins, "Automated sizing, counting and identification of zooplankton by pattern recognition," *Marine Biology*, vol. 78, pp. 329–334, 1984.
- [30] M. Mayo and A. T. Watson, "Automatic species identification of live moths," *Know.-Based Syst.*, vol. 20, no. 2, pp. 195–202, 2007.
- [31] M. T. Do, J. M. Harp, and K. C. Norris, "A test of a pattern recognition system for identification of spiders," *Bulletin of Entomological Research*, vol. 89, pp. 217–224, 1999.
- [32] K. Russell, M. Do, J. Huff, and N. Platnick, "Introducing spida-web: wavelets, neural networks and internet accessibility in an image-based automated identification system." 2005.
- [33] N. Larios, H. Deng, W. Zhang, M. Sarpola, J. Yuen, R. Paasch, A. Moldenke, D. A. Lytle, S. R. Correa, E. N. Mortensen, L. G. Shapiro, and T. G. Dietterich, "Automated insect identification through concatenated histograms of local appearance features," *8th IEEE Workshop on Applications of Computer Vision (WACV'07)*, 2007.
- [34] E. N. Mortensen, E. L. Delgado, H. Deng, D. Lytle, A. Moldenke, R. Paasch, L. Shapiro, P. Wu, W. Zhang, and T. G. Dietterich, "Pattern recognition for ecological science and environmental monitoring: An initial report," In N. MacLeod and M. O'Neill (Eds.), 2005.

- [35] P. J. D. Weeks, I. D. Gauld, K. J. Gaston, and M. A. O'Neill, "Automating the identification of insects: a new solution to an old problem," *Bulletin of Entomological Research*, vol. 87, pp. 203–211, 1997.
- [36] P. J. D. Weeks, M. A. O'Neill, K. J. Gaston, and I. D. Gauld, "Automating insect identification: exploring the limitations of a prototype system," *Journal of Applied Entomology*, vol. 123, no. 1, pp. 1–8, 1999.
- [37] B. Arbuckle, S. Schroader, V. Steinhage, and D. Wittmann, "Biodiveristy informatics in action: Identification and monitoring of bee species using abis," in *Proc. 15th International Symposium for Environmental Protection*, Zurich, 2001, pp. 425–430.
- [38] T. J. McKeever, "The estimation of individual fish size using broadband acoustics with free-swimming salmonids," Master's thesis, Memorial University of Newfoundland, 1998.
- [39] W. A. Kuperman and P. Roux, *Acoustic Method of Fish Counting and Fish Sizing in Tanks*, California Sea Grant College Program, University of California, San Diego, 2004.
- [40] C. L. Wyatt, M. Trivedi, and D. R. Anderson, "Statistical evaluation of remotely sensed thermal data for deer census," *The Journal of Wildlife Management*, vol. 44, no. 2, pp. 397–402, 1980.
- [41] C. L. Wyatt, D. R. Anderson, R. Harshbarger, and M. M. Trivedi, "Deer census using a multispectral linear array instrument," in *Proceedings of the 18th International Symposium on Remote Sensing of Environment*, Paris, France, 1984, pp. 1475–1488.
- [42] R. G. Best, R. Fowler, D. Hause, and M. Wehde, "Aerial thermal infrared census of canada geese in south dakota," *Photogrammetric Eng. and Remote Sensing*, vol. 48, pp. 1869–1877, 1984.
- [43] L. L. Strong, D. S. Gilmer, and J. A. Brass, "Inventory of wintering geese with a multispectral scanner," *The Journal of Wildlife Management*, vol. 55, no. 2, pp. 250–259, 1991.
- [44] K. V. Embleton, C. E. Gibson, and S. I. Heaney, "Automated counting of phytoplankton by pattern recognition: a comparison with a manual counting method," *Journal of Plankton Research*, vol. 25, no. 6, pp. 669–681, 2003.
- [45] D. S. Gilmer, J. A. Brass, L. L. Strong, and D. H. Card, "Goose counts from aerial photographs using an optical digitizer," *Wildlife Society Bulletin*, vol. 16, pp. 204–206, 1988.

- [46] D. Bajzak and J. F. Piatt, "Computer-aided procedure for counting waterfowl on aerial photographs," *Wildlife Society Bulletin*, vol. 18, pp. 125–129, 1990.
- [47] D. J. Cunningham, W. H. Anderson, and R. M. Anthony, "An image processing program for automated counting," *Wildlife Society Bulletin*, vol. 24, pp. 345–347, 1996.
- [48] A. S. Laliberte and W. J. Ripple, "Automated wildlife counts from remotely sensed imagery," *Wildlife Society Bulletin*, vol. 31, pp. 362–371, 2003.
- [49] P. N. Trathan, "Image analysis of color aerial photography to estimate penguin population size," *Wildlife Society Bulletin*, vol. 32, no. 2, pp. 332–343, 2004.
- [50] J. Mills, "Development of an automated image analysis system to detect beluga whales in aerial photographs," Master's thesis, Memorial University of Newfoundland, 2006.
- [51] R. Gosine, L. Gamage, and A. Trites, "Image processing techniques for automatic counting of sea-lions from aerial images: Results from a feasibility study," in *995 IASTED International Conference on Modeling and Simulation*, Colombo, Sri Lanka, 1995.
- [52] C. Barbraud, K. C. Delord, T. Micol, and P. Jouventin, "First census of breeding seabirds between cap bienvenue (terre adielie) and moyles islands (king george v land), antarctica: new records for antarctic seabird populations," *Polar Biology*, vol. 21, pp. 146–150, 1999.
- [53] W. R. Fraser, J. C. Carlson, P. A. Duley, E. J. Holm, and D. L. Patterson, "Using kite-based aerial photography for conducting adielie penguin censuses in antarctica," *Waterbirds*, vol. 22, no. 3, pp. 435–440, 1999.
- [54] M. Grzimek and B. Grzimek, "Flamingoes censused in east africa by aerial photography," *The Journal of Wildlife Management*, vol. 24, no. 2, pp. 215–217, 1960.
- [55] S. Coutourier, R. Courtois, H. Crepeau, L.-P. Rivest, and S. Luttich, "Calving photocensus of the riviere george caribou herd and comparison with an independent census," *Rangifer*, vol. 9, pp. 283–296, 1996.
- [56] A. L. Lovaas, J. L. Egan, and R. R. Knight, "Aerial counting of two montana elk herds," *The Journal of Wildlife Management*, vol. 30, no. 2, pp. 364–369, 1966.
- [57] M. R. M. Otten, J. B. Haufler, S. R. Winterstein, and L. C. Bender, "An aerial censusing procedure for elk in michigan," *Wildlife Society Bulletin*, vol. 21, no. 1, pp. 73–80, 1993.

- [58] J.-F. Gosselin, V. Lesage, and A. Robillard, "Population index estimate for the beluga of the st lawrence river estuary in 2000," Fisheries and Oceans Canada, Maurice-Lamontagne Institute, Research Document 2001/049, 2001.
- [59] A. R. Hiby, D. Thompson, and A. J. Ward, "Census of grey seals by aerial photography," *Photogrammetric Record*, vol. 123, no. 71, pp. 589–594, 1988.
- [60] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.
- [61] *ERDAS Imagine®*, Leica Geosystems Geospatial Imaging, LLC, <http://gi.leica-geosystems.com>, 2006.
- [62] *ImageTool*, UTHSCSA, <http://www.ddsdx.uthscsa.edu/dig/itdesc.html>, 2006.
- [63] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., 1995.
- [64] L. Hogan, G. Mann, and R. Gosine, *Automated Segmentation of Seal Pups from Aerial Images*, Faculty of Engineering, Memorial University of Newfoundland, 2005.
- [65] *Matrox Inspector*, *Interactive Windows® imaging software for scientific and industrial applications*, Matrox Electronics Systems Ltd., [www.matrox.com/imaging](http://www.matrox.com/imaging).
- [66] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. John Wiley & Sons, Inc., 2001.
- [67] T. Y. Young and K. S. Fu, *Handbook of pattern recognition and image processing*. Academic Press, 1986.
- [68] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice-Hall, 2002.
- [69] D. Luo, *Pattern Recognition and Image Processing*. Horwood Publishing Limited, 1998.
- [70] A. R. Weeks, *Fundamentals of Electronic Image Processing*, 1st ed. Wiley-IEEE Press, January 1996.
- [71] B. Bhanu, S. Lee, and S. Das, "Adaptive image segmentation using genetic and hybrid search methods," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 4, pp. 1268–1291, October 1995.
- [72] N. Otsu, "A threshold selection method from gray-level histogram," *IEEE Trans. Systems Man Cybernetics*, vol. 8, pp. 62–66, 1979.

- [73] O. Demirkaya, "Between-class variance as a measure of image bimodality and sharpness," *Proceedings - 19th International Conference - IEEE/EMBS*, pp. 590–593, Oct.30-Nov.2 1997.
- [74] O. Chutatape and B. Dawson, "A thresholding method for blood aggregate images," *IEEE Engineering in Medicine and Biology*, vol. 15, no. 3, pp. 103–108, 1996.
- [75] P. K. Sahoo, S. Soltani, and A. K. C. Wong, "A survey of thresholding techniques," *Computer Vision, Graphics, and Image Processing*, vol. 41, pp. 233–260, 1988.
- [76] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.
- [77] D. M. J. Tax, "One-class classification: Concept-learning in the absence of counter-examples," Ph.D. dissertation, Technical University of Delft, Advanced School for Computing and Imaging, 2001.
- [78] R. Duda and P. Hart, *Pattern Recognition and Scene Analysis*. John Wiley & Sons, New York, 1973.
- [79] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, Walton Street, Oxford OX2 6DP, 1995.
- [80] D. M. J. Tax and K.-R. Müller, "Feature extraction for one-class classification," in *Artificial Neural Networks and Neural Information Processing-ICANN/ICONIP 2003*. Springer-Verlag Berlin Heidelberg, June 2003, pp. 342–349.
- [81] D. Wolpert, *The Mathematics of Generalization*. Goehring D., Santa Fe Institute, 1994.
- [82] M. Moya, M. Koch, and L. Hostetler, "One-class classifier networks for target recognition applications." in *Proceedings world congress on neural networks*. Portland, OR: International Neural Network Society, INNS, 1993, pp. 797–801.
- [83] G. Ritter and M. Gallegos, "Outliers in statistical pattern recognition and an application to automatic chromosome classification," *Pattern Recognition Letters*, vol. 18, pp. 525–539, 1997.
- [84] C. Bishop, "Novelty detection and neural network validation," *IEE Proceedings on Vision, Image and Signal Processing. Special Issue on Applications of Neural Networks*, vol. 141, no. 4, pp. 217–222, 1994.

- [85] N. Japkowicz, "Concept-learning in the absence of counter-examples: an autoassociation-based approach to classification," Ph.D. dissertation, New Brunswick Rutgers, The State University of New Jersey, 1999.
- [86] C. He, M. Girolami, and G. Ross, "Employing optimized combinations of one-class classifiers for automated currency validation," *Pattern Recognition*, vol. 37, pp. 1085–1096, 2004.
- [87] Y. Arzhaeva, D. M. J. Tax, and B. van Ginneken, "Improving computer-aided diagnosis of interstitial disease in chest radiographs by combining one-class and two-class classifiers," in *SPIE Medical Imaging*, J. M. Reinhardt and J. P. W. Pluim, Eds., vol. 6144. SPIE, Bellingham, WA, 2006.
- [88] Y. Tu, G. Li, and H. Dai, "Integrating local one-class classifiers for image retrieval," in *ADMA*, ser. Lecture Notes in Computer Science, X. Li, O. R. Zaïane, and Z. Li, Eds., vol. 4093. Springer, 2006, pp. 213–222.
- [89] C. Lai, D. M. J. Tax, R. P. W. Duin, E. Pekalska, and P. Paclík, "On combining one-class classifiers for image database retrieval," in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, F. Roli and J. Kittler, Eds., vol. 2364. Springer, 2002, pp. 212–221.
- [90] E. J. Spinoso and A. C. P. L. F. de Carvalho, "Combining one-class classifiers for robust novelty detection in gene expression data," in *BSB*, ser. Lecture Notes in Computer Science, J. C. Setubal and S. Verjovski-Almeida, Eds., vol. 3594. Springer, 2005.
- [91] M. W. Koch, M. M. Moya, L. D. Hostetler, and R. J. Fogler, "Cueing, feature discovery, and one-class learning for synthetic aperture radar automatic target recognition," *Neural Networks*, vol. 8, no. 7/8, pp. 1081–1102, 1995.
- [92] L. Nanni, "Experimental comparison of one-class classifiers for online signature verification," *Neurocomputing*, vol. 69, pp. 869–873, 2006.
- [93] L. Tarassenko, P. Hayton, and M. Brady, "Novelty detection for the identification of masses in mammograms," in *Proc. of the Fourth International IEE Conference on Artificial Neural Networks*, vol. 409, 1995, pp. 442–447.
- [94] A. Sachs, C. Thiel, and F. Schwenker, "One-class support-vector machines for the classification of bioacoustic time series," *ICGST International Journal on Artificial Intelligence and Machine Learning, AIML*, vol. 6, no. 4, pp. 29–34, 2006.
- [95] B. Zhang and C. Leung, "Robust face recognition by multiscale-kernel associative memory models based on hierarchical spatial-domain gabor transforms," in

- Seventh IEEE International Conference on Automatic Face and Gesture Recognition (FG 2006), 10-12 April 2006, Southampton, UK.* IEEE Computer Society, 2006, pp. 102–107.
- [96] Z. Zeng, Y. Fu, G. I. Roisman, Z. Wen, Y. Hu, and T. S. Huang, “One-class classification for spontaneous facial expression analysis,” in *FG*. IEEE Computer Society, 2006, pp. 281–286.
  - [97] D. M. J. Tax and R. P. W. Duin, “Combining one-class classifiers,” in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, J. Kittler and F. Roli, Eds., vol. 2096. Springer, 2001, pp. 299–308.
  - [98] C. Metz, “Basic principles of roc analysis,” *Seminars in Nuclear Medicine*, vol. VIII, no. 4, 1978.
  - [99] A. Bradley, “The use of the area under the roc curve in the evaluation of machine learning algorithms,” *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.
  - [100] D. M. J. Tax, *Data Description Toolbox, dd\_tools 1.5.0: A Matlab toolbox for data description, outlier and novelty detection*, Faculty EWI, Delft University of Technology, 2006.
  - [101] V. Barnett and T. Lewis, *Outliers in statistical data*, 2nd ed., ser. Wiley series in probability and mathematical statistics. John Wiley & Sons Ltd., 1978.
  - [102] E. Parzen, “On estimation of a probability density function and mode,” *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.
  - [103] R. P. W. Duin, “On the choice of the smoothing parameters for parzen estimators of probability density functions,” *IEEE Transactions on Computers*, vol. C-25, no. 11, pp. 1175–1179, 1976.
  - [104] M. Kraaijveld and R. Duin, “A criterion for the smoothing parameter for parzen-estimators of probability density functions,” Delft University of Technology, Tech. Rep., 1991.
  - [105] V. Vapnik, *Statistical Learning Theory*. Wiley, 1998.
  - [106] D. Tax, D. de Ridder, and R. P. W. Duin, “Support vector classifiers: a first look,” in *Proceedings of the Third Annual Conference of the Advanced School for Computing and Imaging, ASCI*, Delft, Netherlands, June 1997, pp. 253–258.
  - [107] L. Manevitz and M. Yousef, “One-class svms for document classification,” *J. Mach. Learn. Res.*, vol. 2, pp. 139–154, 2002.
  - [108] T. Hastie and R. Tibshirani, “Discriminant analysis by gaussian mixtures,” *Journal of the Royal Statistical Society series B*, vol. 58, pp. 158–176, 1996.

- [109] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *5th Berkeley symposium on mathematical statistics and probability*, 1967, pp. 281–298.
- [110] G. Strang, *Linear algebra and its applications*. Harcourt Brace Jovanovich College Publishers, 1988.
- [111] S. Abe, *Support Vector Machines for Pattern Classification*. Springer-Verlag London Limited, 2005.
- [112] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.
- [113] A. Smola, B. Schölkopf, and K. Müller, "The connection between regularization operators and support vector kernels," *Neural Networks*, vol. 11, pp. 637–649, 1998.
- [114] D. M. J. Tax and P. Juszczak, "Kernel whitening for one-class classification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 17, no. 3, pp. 333–347, 2003.
- [115] D. M. J. Tax and R. P. W. Duin, "Support vector domain description," *Pattern Recognition Letters*, vol. 20, no. 11-13, pp. 1191–1199, December 1999.
- [116] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2002.
- [117] J. E. Jackson, *A Users' Guide to Principal Components*. John Wiley & Sons, Inc., 1991.
- [118] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as kernel eigenvalue problem," *Neural Computing*, vol. 10, no. 5, pp. 1299–1319, 1998.
- [119] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, and D. Tax, *PRTTools4, A Matlab Toolbox for Pattern Recognition*, Delft University of Technology, 2004.
- [120] G. Giacinto, R. Perdisci, and F. Roli, "Network intrusion detection by combining one-class classifiers," in *ICIAP*, ser. Lecture Notes in Computer Science, F. Roli and S. Vitulano, Eds., vol. 3617. Springer, 2005, pp. 58–65.
- [121] E. Pekalska, M. Skurichina, and R. P. W. Duin, "Combining dissimilarity-based one-class classifiers," in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, F. Roli, J. Kittler, and T. Windeatt, Eds., vol. 3077. Springer, 2004, pp. 122–133.



- [122] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Transaction on Information Theory*, pp. 179–187, Feb 1962.

# Appendix A

## Pseudocode for ICC Algorithm

---

**Algorithm 1** Isolate Connected Components

---

**Input:** Threshold image  $I_T$

**Output:** Label matrix identifying objects

$r \leftarrow 2$  //initialize object labels; 1 is used to label perimeter pixels

$N \leftarrow$  number of 4-connected objects in threshold image  $I_T$

$SE \leftarrow$  disk-shaped structuring element with radius 2

**for**  $i = 1$  to  $N$  **do**

$O_i \leftarrow$   $i$ th 4-connected object

$O_i \leftarrow$  apply morphological closing to  $O_i$  to remove holes 1 pixel in size

**if**  $\text{area}(O_i) \geq 230$  **then**

$P_i \leftarrow$  perimeter of  $O_i$

        remove perimeter  $P_i$  to divide  $O_i$  into component objects

$M \leftarrow$  number of 4-connected component objects

**for**  $j = 1$  to  $M$  **do**

$O_{ij} \leftarrow$   $j$ th component object of  $O_i$

**if**  $\text{area}(O_{ij}) < 15$  **then**

                remove  $O_{ij}$

**else if**  $\text{area}(O_{ij}) \geq 230$  **then**

$P_{ij} \leftarrow$  perimeter of  $O_{ij}$

                remove perimeter  $P_{ij}$  to divide  $O_{ij}$  into subcomponent objects

$L \leftarrow$  number of 4-connected subcomponent objects

**for**  $k = 1$  to  $L$  **do**

$O_{ijk} \leftarrow$   $k$ th subcomponent object of  $O_{ij}$

**if**  $\text{area}(O_{ijk}) < 10$  **then**

                        remove  $O_{ijk}$

**else**

                        label object  $O_{ijk}$  as  $r$

                        reattach  $P_{ij}$  to  $O_{ijk}$  using labeling and 8-connected rule

                        reattach  $P_i$  to  $O_{ijk}$  using labeling and 8-connected rule

                        apply morphological closing to  $O_{ijk}$  using  $SE$

$r \leftarrow r + 1$

**end if**

**end for**

**else**

                label object  $O_{ij}$  as  $r$

                reattach  $P_i$  to  $O_{ij}$  using labeling and 8-connected rule

                apply morphological closing to  $O_{ij}$  using  $SE$

$r \leftarrow r + 1$

**end if**

**end for**

**else**

        label object  $O_i$  as  $r$

        apply morphological closing to  $O_i$  using  $SE$

$r \leftarrow r + 1$

**end if**

**end for**

---

## Appendix B

### Performance of Segmentation Algorithm on Training Data

The following sections give several examples of how the segmentation algorithm performed on training data. Performance is categorized as excellent, good, satisfactory, poor, or failure. The criteria for each category is given at the beginning of each section. In the example grayscale images shown below, the seal pup is circled.

#### B.1 Excellent Performance

Segmentation results are evaluated as “excellent” if the algorithm cleanly segments 100% of the seal pup with no attached artifacts, such as shadows or ice. Table B.1 shows several examples.

Table B.1: Examples of segmentation performance evaluated as “excellent”.











Subimage	Segmentation Result
	
Continued on next page	











Table B.1 – continued

Subimage	Segmentation Result
	
	
	
	

## B.2 Good Performance

Segmentation results are evaluated as “good” if the algorithm segments at least 90% of the seal pup. A very small part of the seal may be segmented as background and/or small artifacts, such as shadows or ice, may be segmented as part of the seal object; this slightly distorts the true shape of the seal. Table B.2 shows several examples.

Table B.2: Examples of segmentation performance evaluated as “good”.

Subimage	Segmentation Result
	
	
	
	
	

### B.3 Satisfactory Performance

Segmentation results are evaluated as "satisfactory" if the algorithm segments at least 50% of the seal pup. A portion of the seal object may be segmented as background and/or artifacts, such as shadows or ice, may be segmented as part of the seal object; the true shape of the seal is still apparent. Table B.3 shows several examples.

Table B.3: Examples of segmentation performance evaluated as "satisfactory".











Subimage	Segmentation Result
	
	
	
Continued on next page	

Table B.3 – continued

Subimage	Segmentation Result
	
	

## B.4 Poor Performance

Segmentation results are evaluated as “poor” if the algorithm segments a very small portion of the seal pup ( $< 50\%$ ), or the seal is adjoined to other objects (e.g. adjacent ice) which are segmented together as a single object; this does not accurately represent the true shape of the seal. Table B.4 shows several examples.

Table B.4: Examples of segmentation performance evaluated as “poor”.



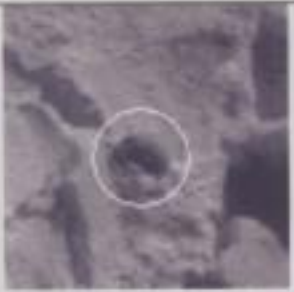

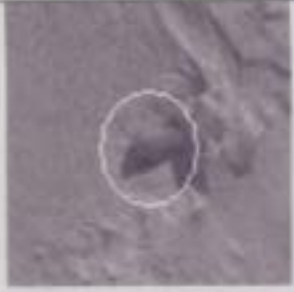




Subimage	Segmentation Result
	
Continued on next page	













Table B.4 – continued

Subimage	Segmentation Result
	
	
	
	

## B.5 Failure

Segmentation results are evaluated as “failure” if the algorithm segments 0% the seal pup. Table B.5 shows several examples.

Table B.5: Examples of segmentation performance evaluated as “failure”.

Subimage	Segmentation Result
	
	
	
	
	



# Appendix C

## Hu Moments

The seven moment invariants,  $\phi_i, i = 1 \dots 7$ , defined in Section 3.5, are often referred to as Hu moments [122]. This set of moments is invariant to translation, rotation, and scale change. The Hu moments are derived from the second and third normalized central moments [68]. For a digital image  $I$ , the moment of order  $(p + q)$  is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y) \quad (\text{C.1})$$

where  $p, q = 0, 1, 2, \dots$  and  $I(x, y)$  is the intensity value at location  $(x, y)$ . The *central moments* are defined as

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \quad (\text{C.2})$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{and} \quad \bar{y} = \frac{m_{01}}{m_{00}}.$$

The *normalized central moments*, denoted  $\eta_{pq}$ , are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad (\text{C.3})$$

where

$$\gamma = \frac{p + q}{2} + 1 \quad (\text{C.4})$$

for  $p + q = 2, 3, \dots$

# Appendix D

## Classification Theory

### D.1 Error Definitions

As discussed in [77], the total error of function  $f$  on a set of independently distributed training objects is decomposed as:

$$\mathcal{E}(f, \mathbf{w}, \mathcal{X}^{tr}) = \frac{1}{N} \sum_i \varepsilon(f(\mathbf{x}_i; \mathbf{w}), y_i) \quad (\text{D.1})$$

where different definitions for the error function  $\varepsilon$  are possible, depending on the type of  $f(\mathbf{x}_i; \mathbf{w})$ . For a discrete valued  $f(\mathbf{x}_i; \mathbf{w})$ , the 0-1-loss error is used. This counts the number of wrongly classified objects:

$$\varepsilon_{0-1}(f(\mathbf{x}_i; \mathbf{w}), y_i) = \begin{cases} 0, & \text{if } f(\mathbf{x}_i; \mathbf{w}) = y_i, \\ 1, & \text{otherwise.} \end{cases} \quad (\text{D.2})$$

For real-valued functions  $f(\mathbf{x}_i; \mathbf{w}) \in [-1, 1]$ , common error definitions include the mean squared error (MSE):

$$\varepsilon_{\text{MSE}}(f(\mathbf{x}_i; \mathbf{w}), y_i) = (f(\mathbf{x}_i; \mathbf{w}) - y_i)^2 \quad (\text{D.3})$$

and the cross entropy (where the labels should be rescaled to positive values  $y_i = \{0, 1\}$ ):

$$\varepsilon_{\text{ce}}(f(\mathbf{x}_i; \mathbf{w}), y_i) = f(\mathbf{x}_i; \mathbf{w})^{y_i} (1 - f(\mathbf{x}_i; \mathbf{w}))^{1-y_i}. \quad (\text{D.4})$$

### D.2 SVDD with Negative Examples

Negative examples (objects which should be rejected) can be used to improve the SVDD by defining a tighter boundary around the data in the areas where outlier objects are present. In contrast with the target examples, which should be within the hypersphere, the negative examples should be outside it.

For the following derivation (summarized from Tax [77]), target objects are enumerated by indices  $i, j$  and outlier objects are enumerated by  $l, m$ . In addition, target objects are labeled  $y_i = 1$  and outlier objects are labeled  $y_i = -1$ . Again, slack variables  $\xi_i \geq 0$  and  $\xi_l \geq 0$  are introduced to allow for errors in the target and outlier sets:

$$\mathcal{E}(R, \mathbf{a}, \boldsymbol{\xi}) = R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l \quad (\text{D.5})$$

and the constraints:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \quad \|\mathbf{x}_l - \mathbf{a}\|^2 > R^2 - \xi_l, \quad \xi_i \geq 0, \xi_l \geq 0 \quad \forall i, l \quad (\text{D.6})$$

Objects with  $\xi_i > 0$  are the false negatives and objects with  $\xi_l > 0$  are false positives. Incorporating the above constraints into Equation (D.5) and introducing Lagrange multipliers  $\alpha_i, \alpha_l, \gamma_i, \gamma_l$  gives:

$$\begin{aligned} L(R, \mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) &= R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l - \sum_i \gamma_i \xi_i - \sum_l \gamma_l \xi_l \\ &\quad - \sum_i \alpha_i [R^2 + \xi_i - \|\mathbf{x}_i - \mathbf{a}\|^2] - \sum_l \alpha_l [\|\mathbf{x}_l - \mathbf{a}\|^2 - R^2 + \xi_l] \end{aligned} \quad (\text{D.7})$$

with  $\alpha_i \geq 0, \alpha_l \geq 0, \gamma_i \geq 0, \gamma_l \geq 0$ .

The partial derivatives of  $L$  with respect to  $R, \mathbf{a}$  and  $\xi_i$  ( $\xi_l$ ) are set to 0 resulting in the new constraints:

$$\sum_i \alpha_i - \sum_l \alpha_l = 1 \quad (\text{D.8})$$

$$\mathbf{a} = \sum_i \alpha_i \mathbf{x}_i - \sum_l \alpha_l \mathbf{x}_l \quad (\text{D.9})$$

$$0 \leq \alpha_i \leq C_1, \quad 0 \leq \alpha_l \leq C_2 \quad \forall i, l \quad (\text{D.10})$$

When these constraints are substituted into (D.7) we obtain

$$\begin{aligned} L &= \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_l \alpha_l (\mathbf{x}_l \cdot \mathbf{x}_l) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ &\quad + 2 \sum_{l,j} \alpha_l \alpha_j (\mathbf{x}_l \cdot \mathbf{x}_j) - \sum_{l,m} \alpha_l \alpha_m (\mathbf{x}_l \cdot \mathbf{x}_m) \end{aligned} \quad (\text{D.11})$$

Equation (D.11) can be simplified when new variables  $\alpha_i'$  are defined which include the labels  $y_i = \pm 1$ :

$$\alpha_i' = y_i \alpha_i. \quad (\text{D.12})$$

Index  $i$  now enumerates both target and outlier objects. Using  $\alpha_i'$  the SVDD with negative examples becomes identical to the original SVDD (Equation (4.25)). The first two terms in (D.11) collapse to the first term in (4.25) and the last three terms reduce to the second term in (4.25). Then the constraints in (D.8) and (D.9) become  $\sum_i \alpha_i' = 1$  and  $\mathbf{a} = \sum_i \alpha_i' \mathbf{x}_i$ , and the function  $f_{SVDD}$  in Equation (4.28) can be

used. Thus, when outlier examples are available,  $\alpha_i'$  will be used instead of  $\alpha_i$  in the optimization and the evaluation.

One might ask the question: if a SVDD can be trained using two classes (targets and outliers), then why not use a conventional two-class classifier? The choice between a SVDD and an ordinary classifier is influenced by both the number of outlier objects available for training and how well they represent the target and outlier distributions. A conventional classifier distinguishes between two (or more) classes without special focus any of the classes. When a representative sample from the target class and a large amount of example outliers is available, and when it is assumed that these objects are independently drawn from the same target and outlier distributions, an ordinary two-class classification problem is obtained. However, the conventional classifier is expected to perform poorly when just a few outlier examples are available and the outlier class is undersampled. In this case, the SVDD will work better because it obtains a close boundary around the target class without requiring a strict representative sample of the target distribution (some outliers are acceptable). Furthermore, training the SVDD with sample outliers is intended to improve the description by obtaining a tighter boundary around the data in areas where outlier objects are present.

### D.3 Kernel Whitening

The following theory is summarized from Tax and Juszczak [114]. Assume the data  $\mathcal{X}^{tr}$  is mapped to the kernel space  $\mathbb{F}$  by some mapping  $\Phi : \mathbb{R}^d \rightarrow \mathbb{F}$  and that the transformed data is centered in this space, i.e.  $\sum_i \Phi(\vec{x}_i) = 0$ . Now the covariance matrix  $C$  of the mapped data set can be estimated by

$$C = \frac{1}{n} \sum_i \Phi(\vec{x}_i) \Phi(\vec{x}_i)^T \quad (\text{D.13})$$

The eigenvectors  $\vec{v}$  and eigenvalues  $\lambda$  satisfy

$$C\vec{v} = \frac{1}{n} \sum_j (\Phi(\vec{x}_j) \cdot \vec{v}) \Phi(\vec{x}_j) = \lambda \vec{v} \quad (\text{D.14})$$

Equation (D.14) shows that the eigenvectors with non-zero eigenvalue must be in the span of the mapped data  $\{\Phi(\vec{x}_i)\}$ , which means that  $\vec{v}$  can be expanded as

$$\vec{v} = \sum_i \alpha_i \Phi(\vec{x}_i) \quad (\text{D.15})$$

Multiplying Equation (D.14) from the left with  $\Phi(\vec{x}_k)$  and using Equation (D.15) gives

$$\frac{1}{n} \sum_j (\Phi(\vec{x}_k) \cdot \Phi(\vec{x}_j)) \left( \Phi(\vec{x}_j) \cdot \sum_i \alpha_i \Phi(\vec{x}_i) \right) = \lambda \sum_i \alpha_i (\Phi(\vec{x}_k) \cdot \Phi(\vec{x}_i)) \quad \forall k \quad (\text{D.16})$$

Introducing the kernel matrix  $K_{ij} = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j)$ , the coefficients  $\vec{\alpha}$  from Equation (D.15) can be obtained by solving the eigenvalue problem

$$\lambda \vec{\alpha} = K \vec{\alpha} \quad (\text{D.17})$$

For normal kernel-PCA the eigenvectors should be normalized to unit length. Therefore, for each eigenvector  $\vec{v}^k$ , the  $\vec{\alpha}^k$  are rescaled to

$$\lambda_k (\vec{\alpha}^k \cdot \vec{\alpha}^k) = 1 \quad (\text{D.18})$$

Since we assumed the data is centered in  $\mathbb{F}$ , the original kernel matrix must be transformed. Assume  $K$  is the  $n \times n$  kernel matrix of the training data and  $K^{\text{tst}}$  the  $m \times n$  matrix of new data. The centered kernel matrix is computed by

$$\tilde{K} = K^{\text{tst}} - 1_n^* K - K^{\text{tst}} 1_n + 1_n^* K 1_n \quad (\text{D.19})$$

where  $1_n$  is an  $n \times n$  matrix and  $1_n^*$  is an  $m \times n$  matrix, both with all entries  $1/n$  [118]. It is assumed that the kernel matrices will always be centered using (D.19).

When the coefficients of  $\vec{\alpha}$  are obtained, a new object  $\vec{z}$  can be mapped onto eigenvector  $\vec{v}^k$  in  $\mathbb{F}$  by

$$(\hat{z})_k = (\vec{v}^k \cdot \Phi(\vec{z})) = \sum_i \alpha_i^k (\Phi(\vec{x}_i) \cdot \Phi(\vec{z})) = \sum_i \alpha_i^k K(\vec{x}_i, \vec{z}) \quad (\text{D.20})$$

where  $(\hat{z})_k$  means the  $k$ -th component of vector  $\hat{z}$ .

The data can be transformed into a representation with equal variance in each feature direction by slightly adapting the normalization from Equation (D.18). The variance of the mapped data along component  $\vec{v}^k$  is

$$\text{var}(\mathcal{X}^{tr}) = \frac{1}{n} \sum_j (\hat{x}_j)^2 = \frac{1}{n} \sum_j \left( \sum_i \alpha_i^k k(\vec{x}_i, \vec{x}_j) \right)^2 = \frac{1}{n} (\vec{\alpha}^k)^T K K \vec{\alpha}^k \quad (\text{D.21})$$

Using Equation (D.17) the variance is constant for all features when, instead of Equation (D.18), we use the normalization

$$\lambda_k^2 (\vec{\alpha}^k \cdot \vec{\alpha}^k) = 1 \quad \text{for all considered components } k \quad (\text{D.22})$$



The data set  $\hat{\mathcal{X}}^{tr}$ , transformed using the mapping (D.20) with normalization (D.22), can now be used by *any* one-class classifier. The dimensionality  $d'$  of this data set depends on how many principal components  $\bar{v}^k$  are taken into account. All features now have equal variances and the data is also uncorrelated due to the fact that it is mapped onto the principal components of the covariance matrix [114].

### D.3.1 Chernoff Distance

After applying a suitable kernel whitening to the original data, it is sufficient to use a simple one-class classifier on the mapped data. The complexity is then moved from optimizing a classifier to optimizing the pre-processing. In order to avoid a complete model selection for both the kernel whitening and the classifier, the Chernoff distance [114] between the target and outlier class can be used to select parameters for the kernel whitening. This distance is defined as:

$$J_C = -\log \left[ \int_{\bar{z}} p(\bar{z} | \omega_T)^s p(\bar{z} | \omega_O)^{1-s} d\bar{z} \right] \quad (\text{D.23})$$

where  $s$  is free to choose such that  $0 \leq s \leq 1$ ,  $p(\bar{z} | \omega_T)$  is the data distribution of the target objects, and  $p(\bar{z} | \omega_O)$  is the distribution of the outlier objects. Assuming we have two normally distributed classes with means  $\mu_1$  and  $\mu_2$  and covariance matrices  $\Sigma_1$  and  $\Sigma_2$ , the Chernoff distance reduces to:

$$J_C = \frac{1}{2}s(1-s)(\mu_2 - \mu_1)^T[(1-s)\Sigma_1 + s\Sigma_2]^{-1}(\mu_2 - \mu_1) + \frac{1}{2} \log \left[ \frac{|(1-s)\Sigma_1 + s\Sigma_2|}{|\Sigma_1|^{1-s}|\Sigma_2|^s} \right]. \quad (\text{D.24})$$

In one-class classification it can be assumed that the mean of the outlier class is very close to the mean of the target class, such that  $\mu_O - \mu_T = 0$ . In addition, by the kernel whitening, the data is transformed such that  $\Sigma_T = \mathcal{I}$ , the identity matrix. Taking these last two details into consideration and setting  $s = \frac{1}{2}$ , the Chernoff distance becomes:

$$J_C = \frac{1}{2} \log \left[ \frac{|\mathcal{I}/2 + \Sigma_O/2|}{|\Sigma_O|^{1/2}} \right], \quad (\text{D.25})$$

where  $\Sigma_O$  is the covariance matrix of the outlier class. The Chernoff distance is relatively cheap to compute, and thus the expensive optimization of the one-class classifier and the computation of the AUC for all combinations of kernel definition and data dimensionality can be avoided. For each kernel definition, a near optimal dimensionality can be estimated by plotting the Chernoff distance for varied dimensionalities. Only for these combinations of kernel and dimensionality do the one-class classifier and AUC have to be computed and compared.







